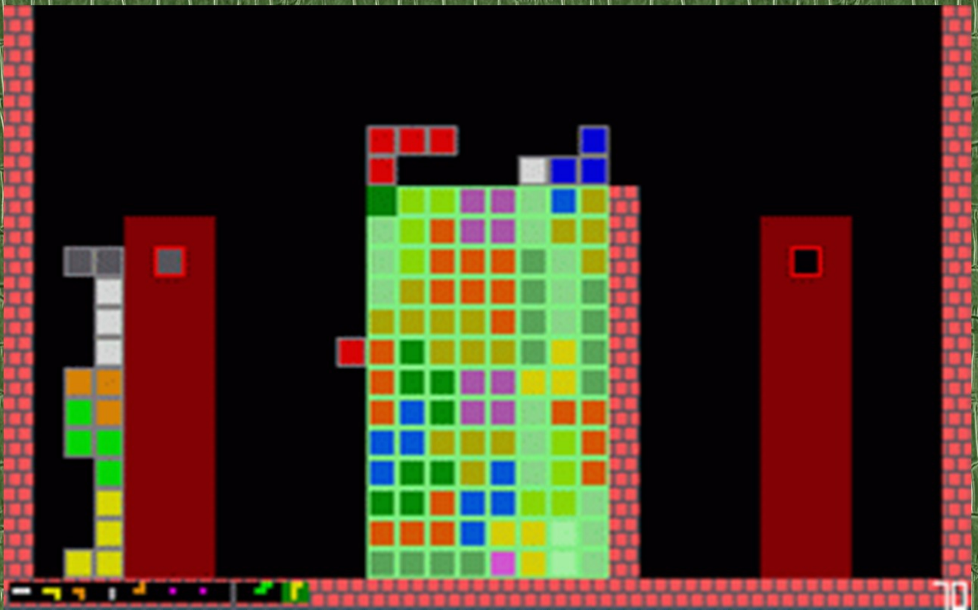
Láng Attila D.



Így készül a program

Láng Attila D.

Így készül a program

2021

Tartalom

Bevezető

Ez a könyvecske 2021-es dátumot visel, de valójában még 2004–2005-ben írtam. Egy fejezet lett volna egy nagyobb könyvben, a BASIC nyelvről írt monográfiában, amit végül nem fejeztem be – ahogy ezt a fejezetet sem, így ez a könyvváltozat is csonkán maradt. Mégis azt hiszem, elég érdekes ahhoz, hogy kiadjam önálló könyvként.

Pár szót a könyvről. Az eredeti mű több részből állt. Az első fejezet elmesélte a BASIC nyelv történetét – ezt a részt felhasználtam *TV–BASIC* című könyvem újrakiadásában, amelynek munkálatai közben döntöttem el, hogy ezt a fejezetet kiadom külön könyvként. A második fejezet egy BASIC-nyelvtanfolyamot tartalmazott, persze szintén befejezetlenül – ezt eldobtam, hiszen a *TV–BASIC* pontosan egy BASIC nyelvtanfolyam volt, amelyből én írtam könyvet. A negyedik fejezet pedig egy átfogó BASIC szótár szeretett volna lenni, amely a BASIC összes fellelhető nyelvjárásának összes kulcsszavát felsorolja, mindegyik szónak az öszszes nyelvjárásbeli jelentésével és minden szükséges tudnivalóval. Néhány nyelvjárás anyaga csakugyan belekerült, de az anyag túlságosan óriási volt ahhoz, hogy egymagam boldoguljak vele.



Az egykori könyv harmadik fejezetét tartja kezében, illetve látja képernyőjén az olvasó. *Így készül a program.* Hasonlót már alkotott Frank Dacosta, akinek könyve sokaknak ismerős, akik az 1980-as években foglalkoztak programozástechnikával. De van egy nagy különbség. Ő megírta a programot előre, aztán írt egy könyvet arról, hogy hogyan írta meg a programot. Az én könyvem egészen másképpen készült: egyszerre írtam a könyvet és a programot. Így az olvasó velem együtt gondolkodhat, amikor eltervezzük, hogy milyen funkciót tegyünk bele következőnek és az hogy működjön, velem együtt törheti a fejét, amikor a programban hiba lép fel, és velem együtt lélegezhet föl, amikor a hibát végre sikerül kiküszöbölni.

Tizenhét év telt el, amióta ezt a programot és könyvet megírtam. Azóta a számítástechnika világa sokat változott. De a QBASIC nyelv, amit használtam, ma is letölthető a netről, és egy DOSBox segítségével ma is futtatható. Az olvasó ma is kipróbálhatja mindazt, amit együtt alkotunk.

Láng Attila D.
2021. június

Bevezető (2004-ből)

Mialatt könyvemen dolgoztam, be kellett látnom, hogy egy BASIC-tanfolyam nem ad teljes áttekintést a programozással ismerkedő Olvasó számára erről az eleinte rejtélyesnek tűnő tevékenységről. Igen, a tanfolyamok leírják, hogyan kell elágazást, ciklust, tömböt létrehozni, hogyan kell *Ó, IÓ, CIÓ, ÁCIÓ, KÁCIÓ, AKÁCIÓ, VAKÁCIÓ* típusú szópiramist kiíratni, de ebből nem derül ki, hogyan áll össze egy program. Egyáltalán hogy kezdjünk hozzá?

Nem éreztem volna teljesnek művemet, ha nem teszek kísérletet a programozó munkájának bemutatására, ámbár ez már nem függ össze szorosan a BASIC nyelvvel: a más nyelven dolgozó programozó is ugyanígy dolgozik, csak másképpen írja ugyanazokat az utasításokat. (Azért mégsem teljesen ugyanígy. A nyelvek között a puszta szintaktikus különbségeken túl szemléletbeli különbségek is vannak.) A programozásról még mit sem tudó Olvasó nem kaphatott volna teljes képet a BASIC-ről, ha nem látja gyakorlatban, használat közben is. Le is ülhet persze programozni, és az *igazán* teljes képet valóban csak így kapja meg.

Morfondír

Ez a fejezet arról szól, hogyan dolgozik a programozó. A nehezebbik végéről fogtam meg a dolgot. Könnyű lett volna úgy megírni ezt a fejezetet, hogy veszek egy kész programot, aztán lépésről lépésre leírom, hogy mi mit csinál benne, de ebből az Olvasó nem tudta volna meg, hogy a program hogyan állt össze azzá, aminek kész formájában látja. A program nem úgy készül, hogy a programozó leírja a legelső sort, aztán a másodikat, a harmadikat, s így tovább szép sorjában, amíg le nem írta az összes sort, akkor lefordítja és kész. Dehogyis. A programlista az utasításokat abban a sorrendben tartalmazza, ahogy a gépnek végre kell hajtani őket, nem pedig abban a sorrendben, ahogy a programozónak eszébe jutottak. Ez utóbbi sorrend a kész programlistáról már kinyomozhatatlan.

E fejezet megírásának nehezebbik, de egyedül célravezető módja az volt, hogy nekiláttam, helyesebben *neki fogok látni* egy program megírásának, s az Olvasót folyamatosan beavatom az eseményekbe. Nemcsak abba, hogy mit írok a BASIC szövegszerkesztőjébe, hanem abba is, hogy miért írom azt. Amint az Olvasó tapasztalni fogja, a programozásnak csupán „külső megjelenése” az, amit a programozó leír és a mellette ülő megfigyelő elolvashat a képernyőn; a munka 99%-a a programozó fejében történik, ahol eldől, hogy *miért* éppen az fog a szövegszerkesztőbe kerülni, ami.

Az imént jövő időben írtam, hogy *neki fogok látni.* Csakugyan: amikor ezeket a sorokat írom, még egyetlen betű sincs megírva abból a programból, amit rövidesen elkészítek az Olvasó szeme láttára. Tudom, mit fogok írni, tudom, hogyan fogom írni, tehát a munka egy részét már azelőtt elvégeztem, hogy hozzákezdtem e

fejezet megírásához, de az Olvasó így sem marad le semmiről, mindenbe be fogom avatni.

Azt mondtam, a programozói munka 99%-a fejben történik. De mi is ez a munka? Egyszerűen *morfondírozás.* A programozó úgy dolgozik, hogy végiggondol valamit: *„Ezt és ezt az eredményt szeretném elérni. Hogy csináljam? Erre a dologra kétféle módszert ismerek. Az első módszer így és így működik, ez és ez az előnye, amaz és amaz a hátránya. A másik módszerrel emígy és emígy áll a dolog. Azt hiszem, célszerűbb lesz, ha az első módszert választom, mert ha a másodikat használom, akkor emennél meg amannál a műveletnél problémák adódhatnak. Az első módszert fogom használni."* Miután ezt végiggondolta, nekiül és megír egy programrészt, nem ritkán olyan sebességgel nyomkodva a billentyűket, mintha diktálna neki valaki, és semmi kigondolnivalója nem lenne. Tulajdonképpen ezen a ponton nincs is, hiszen már korábban kigondolta, hogy mit fog írni. Lehet, hogy az előző percben, de az is lehet, hogy napokkal előbb.

A programozó gyakran kerül nehéz helyzetbe. *„Idáig megvagyok, de vajon ezt a részfeladatot hogyan valósítsam meg? Több módszert is ismerek, de ha végiggondolom, mindegyik valahol ütközik az eddig alkalmazott stratégiáimmal. Kénytelen leszek átírni a már megcsinált programrészeket. Lássuk csak, hogyan is fogjak hozzá. Ha az egyes számú módszert választom, akkor ezeket meg azokat az adatokat más struktúrába kell rendeznem, és át kell írnom az összes programrészt, amik foglalkoznak velük. A kettes meg a hármas módszer ennél sokkal fájdalmasabb lenne, úgyhogy az egyest fogom alkalmazni. Írjuk csak át azokat a programrészeket."* S a programozó nekiül és végigmegy a programlistán, lehet, hogy többször is egymás után, és javításokat csinál rajta. Ha valaki ül mellette és nézi, hogy mit csinál, legfeljebb azt láthatja, hogy a javítások valamilyen rendszert követnek, hasonló szövegrészeket hasonló módon ír át, de hogy miért teszi, azt nem látja, mert az a programozó fejében játszódott le, és most, amikor az ujjai mozognak a billentyűkön, már csak a végeredmény látszik.

A programozó mellett ülő megfigyelő sokszor azt látja, hogy a programozó újra meg újra lefuttatja a programját, két futtatás között átír egy-két sort, esetleg csak egy-egy számjegyet valahol a lista közepén – rejtély, hogy miért éppen azokat. Bent a programozó fejében ilyenkor ez játszódik le: *„Miért nem azt csinálja a programom, amit várok tőle? Nézzük csak logikusan. Ahelyett, hogy így meg így cselekedne, ezt meg ezt műveli. Várakozásom és a kapott eredmény között ez és ez a különbség, tehát nekem emígy és amúgy kell megváltoztatnom a kódomat, hogy azt kapjam, amit szeretnék. Rajta, írjuk át. Futtassuk le. Még mindig nem jó, elképesztő. Gondolkodjunk logikusan. Ha ez a dolog itten így és így történik, miközben én azt mondtam, hogy ezt meg amazt akarom, akkor hol lehet a programban bármi, ami megváltoztatja a viselkedését? Vizsgáljunk át mindent, aminek köze lehet ezekhez a részekhez. Talán ez itt? Írjuk át és nézzük meg. Nem, nem ez volt az.*

Akkor ez? Nem, ez sem. Ó, megvan, hát persze, itt volt a hiba. De hülye vagyok, hogy nem vettem egyből észre!”

Ez a könyv nem lenne méltó az ***Így készül a program*** címre, ha csak azt írná le, amit a programozó beleír a programjába. Az csak a programszöveg megírása, a *kódolás,* egy pusztán technikai művelet, ami azért kell, hogy a gép tudomást szerezzen a programozó óhajairól. Az igazi munka nem a kódolás, hanem az, amikor a programozó kialakítja óhajait, bent a fejében.

Jó gondolatolvasást kívánok az Olvasónak.

Föl meg le

A klasszikus programozási tankönyvek kétféle módszert ismernek egy program összeállítására.

Az első a *top-down,* vagyis „fentről le” módszer, ami úgy néz ki, hogy vesszük az elvégzendő feladatot és felosztjuk bizonyos számú részfeladatra. A részfeladatokat tovább osztjuk rész-részfeladatokra, majd ezeket is felosztjuk, és így tovább, egészen addig, amíg csupa olyan rész-rész-részfeladatunk marad, amiket már egyszerűen el tudunk magyarázni a számítógépnek. Ekkor először is írunk egy programrészt (a főprogramot), ami az elsőrendű részfeladatokat szép sorban végrehajtja. Aztán megírjuk ezeknek a részfeladatoknak a programrészeit, a másodrendűekét is, és így tovább, amíg minden részfeladatra meg nem írtuk az azt megvalósító programrészt. A program ekkor kész.

Engem ez a módszer arra emlékeztet, amit Péter Rózsa mesél ***Játék a végtelennel*** című kitűnő matematikakönyvében arról, hogyan fog a matematikus oroszlánt, *ha az oroszlán nyugalomban van.* Készítsünk egy alul nyitott ketrecet, ami akkora, hogy elfér benne az oroszlán. Most osszuk föl a Szaharát két egyenlő részre. Legalább az egyikben benne lesz az oroszlán (tudniillik ha a határvonalon heverészik, akkor mind a kettőben benne van). Most vegyünk egy ilyen fél-Szaharát és osszuk két egyenlő részre. Legalább az egyikben benne van az oroszlán. Folytassuk ezt a felezgetést egészen addig, amíg egy-egy ilyen rész kisebb nem lesz, mint a ketrec alapterülete, és még ebben is benne van az oroszlán. Helyezzük rá ketrecünket: megfogtuk az oroszlánt.

Nem tehetek róla, de ha a top-down programozási módszerről hallok, mindig eszembe jut az is, amivel Péter Rózsa folytatja az oroszlánfogásos történetet, arra az esetre, *ha az oroszlán mozog.* „Akkor ez a módszer nem alkalmazható. Pont.” Bizony a gyakorlatban, amikor tényleg meg kell írni egy programot, ki szokott derülni, hogy az oroszlánunk nagyon is mozog, és ez a módszer nem alkalmazható. Pont.

A másik módszer a *bottom-up,* „lentről fel” nevet viseli. Ez így működik: vegyünk egy részfeladatot, ami elő fog fordulni a programunkban, és elég egyszerű

ahhoz, hogy egymagában meg tudjuk oldani. Írjuk meg a hozzá szükséges programrészt. Vegyük a következő részfeladatot és írjuk meg a szükséges programrészt. Ha minden részfeladatunk elfogyott, akkor rakjuk össze a programrészeket működő egésszé.

Ez a módszer engem azokra a játékokra emlékeztet, amiket gyerekújságokban lehet találni. Egy csomó pont van a papíron szanaszét, mellettük számok, és ha a pontokat a számok sorrendjében összekötik, akkor egy értelmes rajz jön ki. Bottom-up programozási módszerrel ilyenformán lehetne készíteni egy ilyen rajzot: *„Rajzolj egy pontot és írd mellé: 1. Rajzolj még egy pontot és írd mellé: 2. Ezt ismételd egészen (mondjuk) 100-ig. Ha minden pontot megrajzoltál, akkor rendezd el őket olyanformán, hogy egy cirkuszi elefánt képét adják, aki dobogón áll és labdát forgat az orrán."* Az az érzésem, hogy egy ilyen leírás alapján elég kevesen tudnának efféle rajzot készíteni.

Érzésem szerint mind a top-down, mind a bottom-up módszert a mérnöki tudományoktól örökölte a számítástechnika. A mérnök a következőképpen készít mozdonyt. Tudja, milyen energiákat milyen mozgásokká kell átalakítani, és tudja, hogy ezt hogyan teheti meg. Ezért megtervez egy sor berendezést, amelyek a mozgások átalakításának bizonyos részműveleteit végzik el, tehát top-down módszerrel dolgozik. Miután a mozdony tökéletesen összeállt papíron, készít egy alkatrészlistát és megveszi vagy legyártatja a szükséges alkatrészeket. Aztán odaállít egy csomó munkást az alkatrészekkel zsúfolt polcok elé, a művezető kezébe nyomja a műszaki rajzot és hazamegy. Mire másnap visszatér, a munkások bottom-up módszerrel fölépítették a mozdonyt, neki csak annyi a dolga, hogy ünnepélyesen elfordítsa az indítókart.

Ha az Olvasó kicsit járatos a mechanikai tudományokban, avagy olvasott vasúttörténeti műveket, akkor tudhatja, hogy azért ez még ott sem megy *ennyire* könnyedén.

Hát a programozástechnikában sem. Előfordulnak természetesen esetek, amikor a programozó (vagy inkább a programozói munkacsoport) pontosan megtervezi előre, hogy a program mit, hogyan, mikor, miért fog csinálni, a legapróbb részletekig kialakítja előre a program felépítését, még mielőtt egy sort is megírnának magából a programból – amikor aztán a kódolásra kerül a sor, akkor tetszés szerint alkalmazhatják akár a top-down, akár a bottom-up módszert, akár a kettő keverékét, mert mindent tudnak előre. Úgy dolgozhatnak, mint az a festő, aki a legapróbb tintapöttyig látja maga előtt a képet, úgyhogy akár azt is megteheti, hogy először is elhelyez egy tucat kék pöttyöt a fal nagyságú hófehér vászon egyes pontjain, aztán két tucat piros pöttyöt különböző helyeken, és már sok ezer pöttyöt elhelyezett, mire kezd valamiféle kép kirajzolódni.

A gyakorlatban azonban, azt hiszem, a legtöbb festő mégis egy arc vagy egy fatörzs kialakításával kezdené. Valami olyan részlettel, ami az egészet meghatározza.

Bentről ki

Jobb híján a „bentről ki”, *inside-out* elnevezést tudom adni annak a programozási módszernek, amit a gyakorlati programozó alkalmaz. A BASIC-programozó pedig gyakorlati programozó, nem a hosszas, elvont tervezgetés embere. Tervezget ő is, hogyne tervezgetne, de egyszerre csak egy-egy részletet gondol ki, azt megcsinálja, kipróbálja, és megtervezi a következő részletet. Először valami fontossal kezdi; ha a programban van egy abszolút legfontosabb részlet, akkor azzal, ha több ilyen van, akkor az egyikkel közülük. A „bentről ki” elnevezést a festmények mintájára adtam: a festő is a legfontosabb részlettel kezdi, és ez általában a kép közepén szokott lenni. Így tehát a kép középső része készül el elsőként (persze még csak vázlatszerűen), aztán más fontos részletek, és végül a kép legkevésbé fontos elemei, amik többnyire kívül vannak.

Elvégre ne feledjük: a BASIC alapelve az, hogy minél gyorsabban eredményt kapjunk. „Mit parancsolsz, édes gazdám?” Elmondjuk, mit szeretnénk, és a BASIC megcsinálja. Aztán kiegészítjük munkánkat további részletekkel, bővítjük és finomítjuk, a BASIC pedig mindent engedelmesen megcsinál.

Kezdjünk hozzá

Mielőtt belefogunk, állapodjunk meg valamiben. QBASIC-ben fogunk programozni, mert így az Olvasó a gyakorlatban is kipróbálhatja az egyes lépéseket, és strukturáltan dolgozhatunk. Sorszámokat persze nem fogunk használni, de hogy ne kelljen minden alkalommal leírni a teljes programot csak azért, hogy az Olvasó tudja, hova kell beillesztenie azt az egy-két sort, „álsorszámokat” fogok írni a sorok elé. A QB képernyőjének jobb alsó sarkában két szám mutatja a sort és az oszlopot, ahol a kurzor tartózkodik: `00001:001`. Az oszlopra nem lesz szükségünk, a sor mondja meg majd, hogy hova tesszük az új sort a programban. Ha például azt írom:

```
5 PRINT
```

akkor az Olvasó tudni fogja, hogy ez azt jelenti: vigye a kurzort föl vagy le a képernyőn addig, amíg a sorszámmutató 5-öt nem mutat, és oda írja be az új sort. Ha egyszerre több sort illesztünk be, számot csak az első elé írok. Ha nem írok semmilyen sorszámot, akkor a lista legvégére kell írni az új sorokat. Persze az új sorok beírásával az alattuk levő sorok más sorszámok alá kerülnek, de ez nem jelent gondot: a sorszámokat mindketten a program pillanatnyilag aktuális állapota szerint (az illető ablakon belül) fogjuk értelmezni.

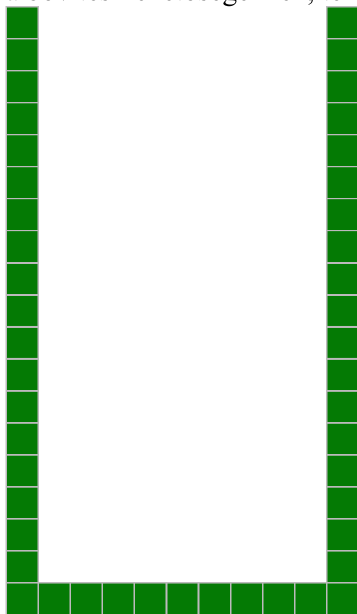
Tetris!

Már majdnem mindent megbeszéltünk, majdnem mindenben megállapodtunk, egyetlen dolog maradt ki. *Mi lesz a program?*

Érett megfontolás után a világ legnépszerűbb számítógépes játékát, a *tetris*t választottam. Aki még nem tetrisezett, az nem játszott számítógépes játékkal. De nem a népszerűsége miatt döntöttem a tetris mellett.

Játékprogramot írni könnyebb, mint felhasználói programot. Szabadabban szárnyalhat a képzeletünk, tetszés szerint vihetjük művünket a legkülönbözőbb irányokba, míg mondjuk egy szövegszerkesztő fejlesztésekor nem csaphatunk le minden kínálkozó ötletre, mert használhatatlanná tesszük vele a programot. (Mondjuk ha eszünkbe jut, hogy milyen jópofa lenne, ha függőlegesen írna vízszintes helyett…) Egy játékprogramnál sokkal kevésbé vannak kötelezően beépítendő szolgáltatások, mint egy felhasználói programnál. Sokkal kevésbé kellemetlen, ha valami nem sikerül tökéletesre. Egyszóval érdekesebb.

A játékprogramokon belül pedig azért választottam a tetrist, mert egyszerű játék, tehát igen gyorsan kaphatunk eredményt; ugyanakkor adódik benne egy-két nem triviális feladat, tehát sokat tanulhatunk belőle; és végtelen tere nyílik benne a bővítési lehetőségeknek, tehát sokféleképpen kísérletezhetünk vele.



Tekintsük át először is, mi lesz a feladat. Adott egy függőleges akna, amiben különböző építőelemek jelennek meg egyesével, és elkezdenek esni. Ilyenek: A játékos esés közben irányítani tudja az elemet: balra-jobbra mozgathatja és 90°-os lépésekben forgathatja. Ha az elem földet ér az akna fenekén vagy a korábban lerakott elemeken, akkor ott marad, többé nem mozdítható. A játékos feladata az, hogy az elemeket úgy rendezze el az aknában, hogy vízszintes sorokat hiánytalanul betöltsön velük; ha ez sikerül, a betöltött sor eltűnik és minden, ami fölötte van, egy sorral lejjebb esik. A játékos veszít, ha az elemekből rakott torony eléri az akna tetejét.

Így szól a tetris játékszabálya az eredeti, Alekszej Pazsitnov által 1985-ben kidolgozott rendszerben. Egyelőre mi sem térünk el tőle, az első feladatunk az lesz, hogy ezt megvalósítsuk hiánytalanul.

Elemgyár

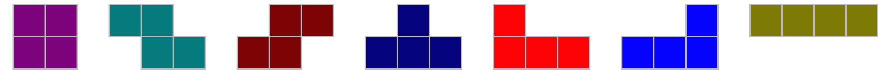
Akkor hát rajta, írjunk tetrist. De mivel is kezdjük?
Morfondírozással, mi mással. Gondoljuk ki, hogyan fogjuk megcsinálni a tetrisünket.

Először is tárolni kell az akna mindenkori állapotát. Mivel az akna valójában egy négyzetrács, egy kétdimenziós tömbben remekül tárolható. Legyen a tömb neve **akna**, egyes elemei pedig jelöljék azt, hogy az aknának azon a pontján mi van: a 0-s érték lesz az üres hely, az 1-es a kocka. A tömb első indexe lesz az *x* koordináta, vagyis az oszlop, a második az *y,* vagyis a sor. Tehát **akna(0,5)** azt fogja megmondani, hogy van-e kocka a legfelső sor közepén. Azzal most még ne foglalkozzunk, hogy hányszor hány kockából fog állni az aknánk. Ez attól fog függeni, hogy mekkora kockákat akarunk elhelyezni a képernyőn, de ez ráér.

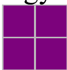
(0,0)	(0,1)	(0,2)	(0,3)	(0,4)	(0,5)	(0,6)	(0,7)	(0,8)	(0,9)
(1,0)	(1,1)	(1,2)	(1,3)	(1,4)	(1,5)	(1,6)	(1,7)	(1,8)	(1,9)
(2,0)	(2,1)	(2,2)	(2,3)	(2,4)	(2,5)	(2,6)	(2,7)	(2,8)	(2,9)

Természetesen az **akna** tömb nem tartalmaz semmilyen adatot arról az elemről, ami éppen esik. Ha annak az adatait is itt helyeznénk el, akkor nem tudnánk megkülönböztetni az eső elemet a korábban leesett elemektől; pontosabban meg lehetne oldani, hogy megkülönböztessük, például 2 lenne a tömbben ott, ahol az eső elem kockái vannak, de ez fölösleges macerát jelentene. Sokkal egyszerűbb, ha az aknát és az eső elemet külön-külön kezeljük.

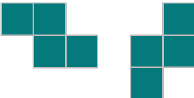
Lássuk ez utóbbit. Elemeink készletét már korábban láttuk:



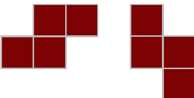
Ezek egyike fog esni lefelé, és nekünk nemcsak mozgatni kell tudni, hanem forgatni is. Persze a legelső minden irányba forgatva ugyanaz marad, de a többi nem. Nézzük csak át a lehetséges állapotaikat, az óramutató járása szerint forgatva. Egyúttal nevet is kaphatnak.



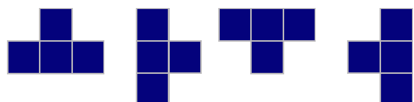
O elem. Nem forgatható.



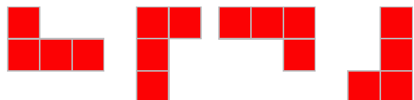
Z elem. Két állapot.



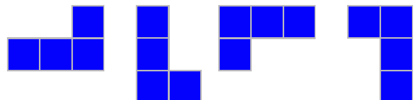
S elem. Két állapot.



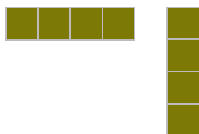
T elem. Négy állapot.



J elem. Négy állapot.



L elem. Négy állapot.



I elem. Négy állapot.

A neveket nem én adtam, ezek a tetriselemek hagyományos elnevezései. Az elmés *LISZTOJ* szó segítségével lehet megjegyezni őket, amit mint a *liszt* és a *pisztoly* összevonását tekinthetünk.

Következzék a morfondír. A kérdés: hogyan tároljuk a tetriselemeket? Több megoldás is kínálkozik.

Mátrixos megoldás. Vegyünk egy 5·5-ös rácsot, amiben 1-esek jelölik a kockákat és 0-k az üres helyeket. Például a **J** elem első állapotát így jelölhetjük:

0 0 0 0 0
0 1 0 0 0
0 1 1 1 0
0 0 0 0 0
0 0 0 0 0

Ha arra vagyunk kíváncsiak, hogy az elem az **x, y** pozícióba állítva ütközik-e az akna tartalmával, csak annyit kell tennünk, hogy végrehajtunk két egymásba ágyazott **FOR** ciklust, mindkettőt 0-tól 4-ig. Legyenek ezek **xx** és **yy**. Most az elemünk megfelelő kockáját **elem(xx, yy)** fogja jelölni, az aknának azt a kockáját pedig, ami vele földrajzilag egybeesik, **akna(x + xx, y + yy)**. Ha tehát **elem(xx, yy) = 1** és **akna(x + xx, y + yy) = 1**, akkor az aknában levő holmik azon a ponton ütköznek az elemünkkel, vagyis az elemet nem szabad odarakni.

A módszer előnye, hogy könnyen ellenőrizni lehet az ütközést, viszont ehhez huszonöt összehasonlítást kell végezni, miközben az elemeink csak négy kockából állnak; a huszonötből huszonegyszer, vagyis az esetek 84%-ában fölöslegesen végezzük el az összehasonlítást. Ráadásul ha később kitalálunk egy olyan elemet, ami kilóg az 5·5-ös rácsból, akkor nagyobb rácsra kell áttérnünk. Ennél biztos tudunk jobbat is.

Koordinátás megoldás. Csak a kockákról tárolunk információt, mégpedig a relatív koordinátáikat. Az egyik kockát kinevezzük középpontnak, ekörül történik a forgatás; ennek a relatív koordinátáit nem is tároljuk, csak a másik hárométt. Például a **J** elem első állapotánál, ha a középponti elem a képen pirossal jelölt kocka (célszerű, hogy az legyen), a következő koordinátákat kell tárolnunk: **0, 0**; **+1, 0**; **-1, 0**; **-1, -1**. (Az első szám mindig az *x,* a második az *y* koordináta.)

Tehát létrehozunk egy háromelemű tömböt, aminek két mezője van: **valami(1).x** fogja jelölni a három kocka közül az első relatív *x* koordinátáját, **valami(1).y** pedig a relatív *y* koordinátát. Az ütközések figyelése most úgy történik, hogy ha a piros kocka az akna **x, y** koordinátájú mezőjén van, akkor végigmegyünk a kockákon 1-től 3-ig, és mindegyiknél megnézzük, hogy **akna(x + valami(i).x, y + valami(i).y) = 1**-e, mert ha igen, akkor ütköztünk. Persze magát a piros kockát is ellenőrizni kell, ezért legjobb, ha felveszünk egy **valami(0)** elemet, aminek **.x** és **.y** mezőjében egyaránt nulla van, és ezt is megnézzük.

A módszerrel igazából nincs komoly baj, egyetlenegyet tudok csak mondani: nem elég rugalmas. Ha később szert teszünk olyan tetriselemekre, amik két vagy öt kockából állnak, akkor be kell vezetnünk egy mutatót a kockák számának nyilvántartására, és ezt külön kezelni.

Ceruzás megoldás. Így neveztem el, mert jobb nevet nem találtam neki. A működése a következő. Képzeljünk el egy ceruzát, ami négy irányba tud mozogni a négyzetrácsban, és tud kockákat rajzolni. A ceruza a piros kockától indul, és olyan parancsokra hallgat, mint *föl, le, balra, jobbra, kocka.* Ez utóbbi hatására kockát rajzol oda, ahol éppen áll. A parancsokat egy szövegben adjuk meg; példaelemünket így a **"kjkbbkk"** szöveg kódolja, ahol a kockákat rajzoló **k** betűket a kocka színére festettem a könnyebb azonosítás végett. (Persze lehetne **"kbkfkjjlk"** is.)

Az ütközéseket most úgy figyeljük, hogy végrehajtjuk a parancsokat, de kocka rajzolása helyett csak megnézzük, hogy azon a helyen van-e akadály az aknában.

A módszer rugalmas és eléggé tömör, a legnagyobb előnyének azonban azt tartom, hogy bármilyen tetriselemet egyetlen szövegben lehet kódolni. Helyesebben négy szövegben, mert a négy elforgatáshoz célszerűen külön-külön szöveget rendelünk hozzá.

A programozó tehát fölvázolta a lehetséges megoldásokat, és elgondolkodott azok előnyein és hátrányain. Végül a *ceruzás megoldás*t választotta, többé-kevésbé azon az elven, ahogyan az egyszeri ember választott feleséget.

– Volt három jelöltem. Mindegyiknek adtam százezer forintot. Az első részvényeket vett rajta, és elmondta, hogy ezzel alapozza meg a közös jövőnket. A második betette a takarékba, hogy később együtt költhessük el közös céljainkra, amikor már kamatozott. A harmadik háztartási gépeket vett, hogy kényelmes legyen az otthonunk.

– És melyiket választottad?
– Micsoda kérdés! Hát amelyiknek a legnagyobb melle van.

Feladatlista

A programozónak mindig van egy listája arról, hogy milyen feladatokat kell elvégeznie. Igaz, ritkán jelenik meg leírva, többnyire csak a programozó fejében található meg. Most azonban írjuk le. Az egyes feladatok kapjanak egy rövidített nevet, majd ezeken emlegetjük őket munka közben.

AKNARAJZ
Az akna és tartalmának kirajzolása.

RAKTÁR
Az összes felhasználható elem ceruzás kódolású tárolása, hogy legyen miből kisorsolni a következőnek leeső elemet.

ELEMRAJZ
Az eső elem kirajzolása.

MOZGÁS
Az eső elem mozgatása, esése, forgatása, az ütközések figyelésével együtt.

SOROK
Az elem leérkezése után annak vizsgálata, hogy betöltöttünk-e egy vagy több sort, és ha igen, azok eltüntetése és a fölöttük levő kockák lejjebb hozatala.

Ezek a legfontosabbak, minden egyéb dolog már másodlagos. Csapjunk hát a kockák közé, kezdődjön a munka. Elindította már az Olvasó a QBASIC-et?

Egy program írását célszerű a **DEFINT A-Z** utasítással kezdeni, vagyis az összes változónkat alapértelmezésként egésszé tenni. Ez kiiktatja a lebegőpontos aritmetikát és meggyorsítja a program futását.

A gyakorlott programozó mindig azzal kezdi a kódolást, hogy létrehoz néhány definiált konstanst, amik a programban használatos fontosabb értékeket rögzítik. Általában azokat az értékeket definiáljuk konstansként, amikről gyanítható, hogy később meg kell változtatni őket, ezért nem a konkrét számot írjuk majd programsorainkba, hanem csak egy nevet, hogy egyetlen utasítás átírásával ki lehessen cserélni a számot. Ilyen érték az elemek száma. Momentán hétféle elemünk van, de semmi sem garantálja, hogy később nem lesznek többen is. Praktikus lesz ezért egy **CONST elemek = 7** utasítás.

Az akna méretei is olyan értékek, amikről nem tudjuk, hogyan fognak alakulni a későbbiekben. Van egy gondolatom: először csináljuk meg a programot karakteres képernyőre, csak aztán térjünk át grafikusra. Kényelmesebb, hamarabb kapunk látható eredményt. A karakteres képernyő 25 sorból és 80 oszlopból áll, ez utóbbi szám azonban túl nagy, elég lesz 12 oszlop. (Ebből kettő a két szélső fal, vagyis tíz oszlopnyi helyünk lesz az elemek rendezgetésére.) Rögzítsük ezt a két számot:

```
CONST aknax = 12, aknay = 25
```

(A tapasztalt programozó már most, három programsor után is megnyomja a *File* menü *Save* parancsát és elmenti munkáját **tetris.bas** néven!)

Most már nyugodtan létrehozhatjuk az akna állapotát tároló tömbünket:

```
DIM akna(1 TO aknax, 1 TO aknay)
```

(Így kényelmesebb hivatkozni rájuk, nem 0-tól **aknax-1**-ig mennek az értékek, hanem 1-től **aknax**-ig.)

Eddig megvolnánk, most nézzük a feladatlistát és válasszunk ki belőle egy feladatot.

Aknarajz

Legyen első kidolgozandó feladatunk az **AKNARAJZ.** Rajzoljuk ki az aknát a lehető legegyszerűbben. A kirajzolást sokszor kell majd elvégezni, ezért rutint írunk rá: *Edit* menü, *New SUB* parancs, névnek beírjuk: **aknarajz**, és megnyomjuk az *Enter*t. Egy ablakot kapunk három sornyi szöveggel:

```
DEFINT A-Z
SUB aknarajz
END SUB
```

A rutin szövegét a **SUB** és az **END SUB** közé kell írnunk.

Ha karakteres képernyőn dolgozunk, akkor elég egyszerű az ügy. Lefuttatunk két egymásba ágyazott ciklust 1-től **aknax**-ig, illetve **aknay**-ig, és minden **akna**-elemről megnézzük, hogy mit tartalmaz. Ha 0-t, akkor kiírunk egy szóközt; ha 1-et, akkor egy inverz szóközt, vagyis tele négyzetet. A QB a DOS jelkészletét használja (hát hiszen DOS-os program), amiben 219-es kódszám alatt van a tele négyzet. Vagyis:

```
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y) = 1 THEN PRINT CHR$(219); ELSE PRINT " ";
NEXT: NEXT
```

Csinálni persze semmit se csinál, hiszen a főprogramból még nem hívtuk meg. Tegyük meg próbaképpen. Ehhez meg kell nyomni az *F2*-t és a listából kiválasztani a **TETRIS.BAS**-t, vagyis a főprogramot. Ennek végére írjuk:

```
aknarajz
```

Már csak le kell futtatni *F5*-tel és máris… közli a BASIC, hogy az **IF** utasításban **Array not defined**, a tömb nincs definiálva. Teljesen igaza van: az **akna** tömböt a főprogramban definiáltuk, lokális változóként, nem osztottuk meg a rutinjainkkal. Márpedig ez egy nagyon fontos tömb, a program számos pontja fog rá hivatkozni, osszuk meg. Menjünk vissza megint a főprogramba és írjuk be a **DIM** utasításba a **SHARED** kulcsszót.

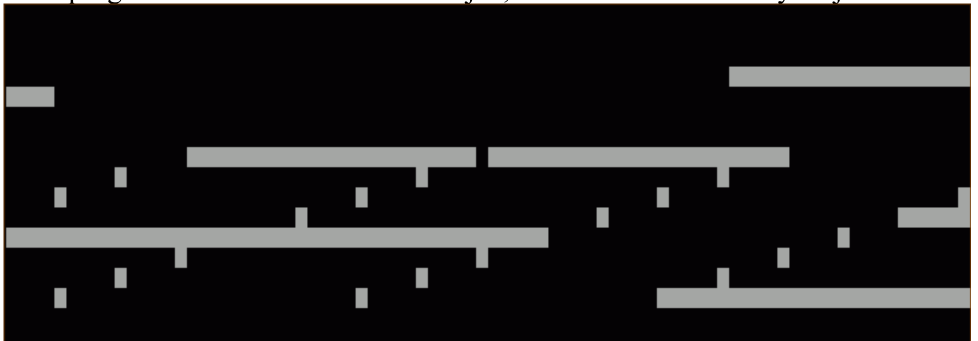
```
DIM SHARED akna(1 TO aknax, 1 TO aknay)
```

Most már lefut a program, csak semmi látható eredményt nem ad. Nem csoda, hiszen az **akna** tömb minden egyes eleme nulla, vagyis kizárólag szóközök jelennek meg. Az akna falait előbb-utóbb úgyis létre kell hozni, hát tegyük meg most, és akkor az **aknarajz** rutint is le tudjuk tesztelni. A főprogramba kerüljön:

```
6 FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT
```

Az első ciklus végigmegy az összes soron a legalsó kivételével, és mindegyik bal és jobb szélső kockájába 1-et tesz. Ezzel megvannak a függőleges falak. A második ciklus az összes oszlopon megy végig, és a legalsó sorba 1-et tesz, ezzel megvan az akna alja is.

A program futtatása… hát mit mondjak, nem a kívánt eredményt adja:



Hol lehet a hiba? Rögtön az elején debugolhatunk egy kicsit. Vagy nem jó helyre tesszük az 1-eseket a tömbben, vagy nem jól íratjuk ki őket. Tegyünk egy próbát. Mi van, ha a függőleges falakat nem rajzoljuk meg a tömbben, csak az akna fenekét? Iktassuk ki a 7. és 8. sort (az elejükre **'**-ot téve), és futtassuk le így.

Egy csomó szétszórt kocka jelenik meg az eddigiek alatt, ezzel végképp áttekinthetetlenné téve a képet. Ugyan tegyünk már be egy **13 CLS** utasítást… (Miért pont oda? Mindegy, hova, csak a rajzolás előtt legyen.)



Az eredmény enyhén szólva gyanús. Tizenkét darab szétszórt kocka, de nem ám akárhogy, hanem szépen emelkedő lépcsőkben. Ez nem lehet véletlen.

A ravasz programozó mindenekelőtt tisztázni szeretne egy-két dolgot. Azt látjuk, hogy hova kerülnek kockák (nem oda, ahova kellene), de azt nem látjuk, hogy hova kerülnek szóközök, mivelhogy a szóköz ugye nem látható. Tegyünk helyette olyasmit, ami látható. Írjuk át az **aknarajz** rutin 5. sorának végét **PRINT "a";**-ra, és futtassuk le így is a programot.

```
aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaa
aaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaa
aaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaa
aaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█aaaaaaaaaaaaaaaaaaaaaaaa█
```

Aha! mondja a programozó nyomatékkal, mint aki mindent ért. (Még nem ért mindent, de minek elárulni?) A program végigírja az első sor mind a nyolcvan oszlopát, holott csak tizenkettőt kellene, aztán a következő sort is végigírja, holott oda is csak tizenkettőt kellene, nem csoda, hogy négy sorra futja csak abból, ami egyébként huszonötre jönne ki. Az aknában 12·25 = 300 kocka van, mi pedig kis híján négy sort írtunk tele; 4·80 = 320, és elég könnyű megszámolni, hogy az utolsó sorban húsz hely maradt üres. Vagyis annyi kockát rajzolunk, amennyit kell, csak egyvégtében. A tizenkettedik kocka után a következő sorban kellene folytatni a kiírást, a huszonnegyedik után megint, hát persze, mondja a programozó, elfelejtettem a sorok végére *Enter* jelet tenni. És gyorsan átírja a rutin 6. sorát így:

NEXT: PRINT: NEXT

```
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
aaaaaaaaaaaaaaaaaaaaaaaa█
```

Már igazán egész jó, ha eltekintünk attól a mellékkörülménytől, hogy nem jó. Most már legalább téglalapformát ölt, a kockák is egyenes vonalban sorakoznak, csak nem vízszintesen, hanem függőlegesen. És egyáltalán, mekkora ez tulajdonképpen? A programozó a képernyőhöz hajol és számolni kezdi az *a* betűket. Egymás alatt sorakozik egy, kettő… tizenkét darab. A programozó gyanút fog és elkezdi vízszintesen számolni őket, abban a szent meggyőződésben, hogy huszonöten lesznek a kockával együtt. Nem is csalódik, tényleg annyian vannak. Vagyis az akna gyönyörűen megjelent, csak el van forgatva 90 fokkal.

Hogy is van ez? A programozó a rutinra mered és gondolkodik.

```
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT "a";
```

```
NEXT: PRINT : NEXT
```

Először **x = 1** és végigmegyünk az **y** cikluson, 1-től 25-ig. Vagyis először **x = 1** és **y = 1**, kirajzoljuk a kockát, aztán **x = 1** és **y = 2**, ez a bal felső sarok alatt levő kocka, kirajzoljuk… no de a **PRINT** nem felülről lefelé ír, hanem balról jobbra! Persze hogy elforgatva jelenik meg, ha felülről lefelé vizsgáljuk a kockák tartalmát, de balról jobbra ábrázoljuk őket. Ugyanabban a sorrendben kell vizsgálni őket, ahogyan ábrázoljuk, az **x = 1**, **y = 1** sarokkocka után a jobb oldali szomszédjának kell következnie, **x = 2**, **y = 1**. És **y**-nak végig 1-nek kell maradnia, mialatt végigmegyünk **x** minden értékén 1-től 12-ig…

A programozó a homlokára csap. Hát persze, föl kell cserélni a két **FOR** utasítást!

```
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT "a";
NEXT: PRINT : NEXT
```

Lefuttatja, s lám, a képernyő bal széle tetrisaknává változik. A programozó boldogan visszacseréli az **a** betűt szóközre és kiveszi az **'**-okat a főprogram két kiiktatott sorából.

Ez a kis hibakeresés ízelítő volt a programozó módszereiből, amiket a felmerülő hibák megoldására használ. Nem tagadom: szándékosan csináltam hibát a programban, kettőt is, hiszen eleve tudtam, hogy szükség van arra a **PRINT**-re a sorok végén, és hogy a két **FOR** ciklus sorrendje korántsem mindegy. De így még egy tanulságot levonhatunk: a hiba abból eredt, hogy az **x** és **y** koordinátákat hagyományosan ábécérendben írjuk, minden geometriakönyvben azt fogja találni az Olvasó, hogy előbb áll az *x,* csak aztán az *y* koordináta. A **PRINT** azonban nem koordinátákkal dolgozik, hanem a latin betűs írás-olvasás hagyományos sorrendjében, vízszintesen halad balról jobbra, aztán felülről lefelé. A **PRINT** lelkivilágában nem *x*-ek és *y*-ok, hanem sorok és oszlopok vannak, ezek logikus sorrendje pedig *sor, oszlop* – nem pedig *oszlop, sor.* Ha **x** és **y** helyett **sor** és **oszlop** (vagy **s** és **o**) változókat használtunk volna, a hiba nem következett volna be, mert amilyen gyanútlanul írtuk most a két **FOR** ciklust ebben a sorrendben, olyan gyanútlanul írtuk volna őket akkor *sor, oszlop,* vagyis *y, x* sorrendben.

Raktár

Folytassuk a munkát a **RAKTÁR** feladattal. Amíg ez nincs kész, a többihez nem kezdhetünk hozzá. A feladat: az elemek tárolása ceruzás kódolásban.

Logikus, hogy a raktár egy tömb legyen, amit a program indulásakor feltöltünk az elemeket kódoló adatokkal, aztán onnan nyúljuk ki őket mindig. Összesen tizenkilencféle elemünk van (a **T, J** és **L** elemeknek négy-négy, a **Z, S** és **I** elemeknek két-két, az **O** elemnek egy állása). De ha ilyen tömböt csinálunk:

(1) az L elem 1. állása
(2) az L elem 2. állása
(3) az L elem 3. állása
(4) az L elem 4. állása
(5) az I elem 1. állása
(6) az I elem 2. állása
(7) az S elem 1. állása

és így tovább, akkor folyton számolgatni kell, hogy az akármelyik elem akármelyik állását tartalmazó tömbelemet kikeressük. Négyesekkel meg kettesekkel kell szorozgatni, fejben tartani, hogy melyik elemnek hány állása van, kinek jó ez?

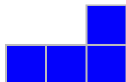
Sokkal jobb lesz, ha kijelentjük, hogy *mindegyik* elemnek négy állása van, és kész. Majd az **O** elemnek mind a négy állapota egyforma lesz.

És ne rakjuk őket egydimenziós tömbbe, hogy ne kelljen szorozgatni. Legyen csak a tömb kétdimenziós. Az egyik index legyen az elem sorszáma, a másik az irányé. Vagyis kell a főprogramba egy **6 DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING** sor, és a továbbiakban **raktar(elem sorszáma, állás sorszáma)** fogja megadni a kívánt elemet. Mármint ha majd föltöltöttük a tömbünket.

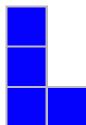
Ha már ennyi mindent végzünk a program elején, az aknát is feltöltjük meg a raktárat is, akkor érdemes csinálni egy külön rutint erre a célra. Így nem a főprogramot nyújtjuk egyre hosszabbra. Csináljunk tehát egy rutint, aminek legyen a neve **inic** (az *inicializálás* szó rövidítéseként), és rakjuk át bele a főprogramból a két **FOR** ciklust. Mögéjük kerülhet a raktár feltöltése az ablak 11. sorától kezdve (betettem 10-esnek egy üres sort).

Hogy is csináljuk? Mondjuk a **DATA** utasítás jó lesz. Rakjuk bele az elemeket, mondjuk a *LISZTOJ* sorrendjében, először az **L** elem mind a négy állását, aztán az **I** mind a négy állását (amiből kettő egyforma), és így tovább. A kódolásnál vigyázzunk arra, hogy elsőként az elem középső kockáját válasszuk, ha van ilyen, mert akkor akörül forgathatjuk majd. Az elsőként megrajzolandó kocka kerül arra a helyre, amit mint az elem pillanatnyi koordinátáit számon tartunk.

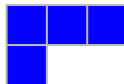
Kezdjünk neki.



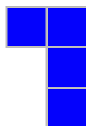
L elem, első állás. Kezdjük a középső kockával, aztán balra, kocka, kettőt jobbra, kocka, fel, kocka. **kbkjjkfk**



Második állás. Középről indulunk, fel, kocka, kettőt le, kocka, jobbra, kocka. **kfkllkjk**



Harmadik állás. Közép, jobbra, kocka, kettőt balra, kocka, le, kocka. **kjkbbklk**



Negyedik állás. Közép, le, kocka, kettőt fel, kocka, balra, kocka. **klkffkbk**

Egy elem kész, tegyük be egy **DATA** sorba a főprogram legvégére.

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk

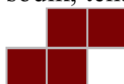


I elem, első állás. A második és a harmadik kocka helyzete teljesen egyforma, kezdjük a másodikkal. Kocka, balra, kocka, kettőt jobbra, kocka, jobbra, kocka. **kbkjjkjk**

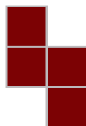


Második állás. Kocka, föl, kocka, kettőt le, kocka, le, kocka. **kfkllklk**

A harmadik állás ugyanaz, mint az első, a negyedik pedig ugyanaz, mint a második, tehát írhatunk egy újabb sort: **DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk**



S elem, első állás. Kezdjük mondjuk a felső sor bal oldali kockájával. Jobbra, kocka, balra, le, kocka, balra, kocka. **kjkblkbk**

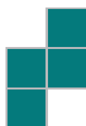


Második állás. A kezdő kockánk most a bal oldali oszlop alsó kockája, föl, kocka, le, jobbra, kocka, le, kocka. **kfkljklk**

A harmadik és a negyedik megint az első kettő ismétlése, kész a harmadik adatsorunk is: **DATA kjkblkbk, kfkljklk, kjkblkbk, kfkljklk**

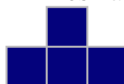


Z elem, első állás. Kezdjük a felső sor jobb oldali kockájával, balra, kocka, jobbra, le, kocka, jobbra, kocka. **kbkjlkjk**

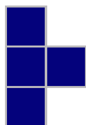


Második állás. A bal oszlop felső kockájával kezdünk, le, kocka, fel, jobbra, kocka, fel, kocka. **klkfjkfk**

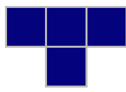
Kész a következő adatsor: **DATA kbkjlkjk, klkfjkfk, kbkjlkjk, klkfjkfk**



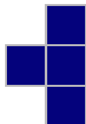
T elem, első állás. A kereszteződésben levő kockával kezdünk, balra, kocka, jobbra, föl, kocka, le, jobbra, kocka. **kbkjfkljk**



Második állás. Kocka, föl, kocka, le, jobbra, kocka, balra, le, kocka. **kfkljkblk**

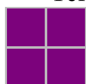


Harmadik állás. Kocka, balra, kocka, jobbra, le, kocka, föl, jobbra, kocka. **kbkjlkfjk**

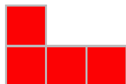


Negyedik állás. Kocka, föl, kocka, le, balra, kocka, jobbra, le, kocka. **kfklbkjlk**.

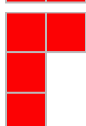
Tehát: **DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk**



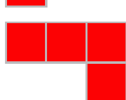
O elem, összes állás. Kezdjük a bal felső sarokban. Kocka, jobbra, kocka, le, kocka, balra, kocka. **kjklkbk**. Ebből annyi lesz, hogy **DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk**



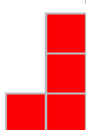
J elem, első állás. A középső kockával kezdünk, jobbra, kocka, kettőt balra, kocka, föl, kocka. **kjkbbkfk**



Második állás. Kocka, le, kocka, kettőt föl, kocka, jobbra, kocka. **klkffkjk**



Harmadik állás. Kocka, balra, kocka, kettőt jobbra, kocka, le, kocka. **kbkjjklk**



Negyedik állás. Kocka, föl, kocka, kettőt le, kocka, balra, kocka. **kfkllkbk**

Kész az utolsó adatsorunk: **DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk**
Idemásolom még egyszer a hét adatsort együtt:
DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, kfkljklk, kjkblkbk, kfkljklk
DATA kbkjlkjk, klkfjkfk, kbkjlkjk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

A programozó nagyon gyakran végez ilyesféle munkát, amilyet most csináltunk: rajzokat, táblázatokat vagy egyéb fajta adatokat tanulmányoz, s kódokat pötyög hosszasan, amik a kívülállónak misztikus, érthetetlen valamik. Legtöbbször fejben végezzük az ilyen átalakítást, előzőleg megtanulva annak módszerét, legfeljebb akkor folyamodunk papírhoz, ha az átalakítás bonyolultabb. Persze ha az Olvasó-

nak így kényelmes, a legegyszerűbb kódolást is végezheti papíron, senki sem tiltja meg.

Kódjaink megvannak, töltsük be őket. Az **inic** rutinba kerüljön a szöveg:

```
11 FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
```

Még annyi a teendőnk, hogy a főprogramba beírjuk az **aknarajz** elé, hogy **inic**, és máris fut a program, szépen kirajzolja az aknánkat megint.

Elemrajz

Következő feladatunk: **ELEMRAJZ**. Egy elem kirajzolása. Lássuk, mit is kell ehhez tudni. Szükség van az elem sorszámára *(LISZTOJ),* állására, és arra, hogy hol van éppen. Mindezeket legjobb lesz a rutinunknak paraméterként átadni.

Készítsünk tehát egy rutint **elemrajz** néven, s miután a QB létrehozta az új ablakot a három sornyi

```
DEFINT A-Z
SUB elemrajz
END SUB
```

szöveggel, írjuk hozzá a **SUB** utasítás végéhez a formális paramétereket:

```
SUB elemrajz (e, i, x, y)
```

E lesz az elem sorszáma, **i** az iránya, **x** és **y** pedig a koordináták.

Mármint a kezdő koordináták, hiszen az elemeink négy kockát foglalnak le, egy koordinátapár viszont egy kocka helyét határozza meg. Az aknának azt a kockáját határozzuk meg **x** és **y** segítségével, ahol az elem ceruzás kódolás szerinti legelső kockája van.

Gondoljuk át, hogy kell egy elemet kirajzolni. Ha a rutin megkapta **e**-t és **i**-t, akkor **raktar(e, i)**-ben meg fogjuk találni azt a ceruzásan kódolt szöveget, ami a kívánt elemet leírja.

```
ceruza$ = raktar(e, i)
```

Menjünk végig a szövegen és nézzük meg, milyen parancsokat tartalmaz.

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
```

Ilyen esetekben a **SELECT CASE** utasítás válik be remekül, ezzel lehet szétválogatni a különböző parancsokat. Kezdjük a mozgásokkal.

```
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
```

Ha a ceruzát mozgató parancsokat találjuk, akkor valamelyik koordinátát megváltoztatjuk, ezzel a mozgás megtörtént. Ha pedig kockarajzoló parancsot találunk, akkor az **x** és **y** által mutatott helyre kirajzolunk egy kockát:

CASE "k": LOCATE y, x: PRINT CHR$(219);

Ide **y, x** kell, hiszen a **LOCATE** sort és oszlopot használ, nem grafikus koordinátákat.

Több parancsunk nem lévén, a munka kész. Zárjuk be az utasításokat.

END SELECT
NEXT

Próbáljuk ki művünket. Mondjuk a főprogramban az **aknarajz** után helyezzünk el egy **elemrajz 1, 1, 3, 5** utasítást: 1. számú elem **(L)** első állását rajzolja ki a 3. oszlopba és az 5. sorba.



Így néz ki az aknának ez a része a kirajzolt elemmel; nem nagyon szép, de a célnak momentán megfelel. Esetleg betehetünk a rutin elejére egy **COLOR 4**, a végére pedig egy **COLOR 7** utasítást, így az elem pirosan jelenik meg és elkülönül az akna szürkéjétől (ez utóbbi fenntartása miatt tettem oda a **COLOR 7**-et, hogy az **elemrajz**ot követő további kiírások megint szürkével dolgozhassanak).

Az Olvasóra bízom, hogy az **elemrajz** rutin hívását kipróbálgassa különböző értékekkel. Arra figyelni kell, hogy ha a koordináták bármikor az elem rajzolása folyamán a képernyőn kívülre esnének, akkor hibaüzenetet kapunk. Ez nem baj, majd gondoskodunk róla, hogy az **elemrajz** a rendeltetésszerű használat során ne kapjon ilyen koordinátákat.

Mozgás

Következik legösszetettebb feladatunk, a **MOZGÁS.** Ez lesz a program legfontosabb rutinja, ami játék közben folyamatosan fut majd és gondoskodik az elem mozgatásáról és a járulékos műveletek végrehajtásáról. Kezdjük azzal, hogy a főprogramból vegyük ki az **elemrajz** próbaképpeni hívását és tegyük a helyére a **mozgas**ét, ami egy új rutin lesz:

DEFINT A-Z
SUB mozgas
END SUB

És lássunk neki.

Az bizonyos, hogy a rutinunkba kell egy ciklus, mert a mozgást sokszor kell végrehajtani. Méghozzá **DO … LOOP** ciklus, egyelőre kilépési feltétel nélkül, ugyanis a kilépési feltétel az lesz, ha valamiért abba kell hagyni a játékot. Tehát tegyünk a rutinba egy **DO** és egy **LOOP** sort, és a továbbiakat írjuk a kettő közé.

Ideje elmorfondírozni, mik legyenek a továbbiak. Az elem ott van az akna tetején… igaz is, az elem még nincs ott az akna tetején. Mielőtt elkezdjük mozgatni, oda kell tenni. Egyelőre tegyünk a **DO** elé egy **e = 1: i = 1: x = 6: y = 2** sort, ami-

ben ugyanazokat az adatokat rögzítjük, amikkel korábban elemet rajzoltunk; és mielőtt bármit művelnénk, rajzoljuk ki az elemet **5 elemrajz e, i, x, y** utasítással, vagyis már a cikluson belül. Merthogy azt mindig ki kell majd rajzolni.

```
SUB mozgas
e = 1: i = 1: x = 6: y = 2
DO
elemrajz e, i, x, y
LOOP
END SUB
```

Csináljuk meg a balra-jobbra mozgatást. Ehhez meg kell néznünk, hogy a felhasználó milyen billentyűt nyomott meg, és ha a 🡄 nyilat, akkor balra vinni az elemet, ha a 🡆-t, akkor jobbra. Ebben az **INKEY$** függvény lesz segítségünkre: **6 a$ = INKEY$**, a QB pedig speciális kódokkal jelzi a billentyűket. Ha a 🡄 nyilat nyomták le, akkor a függvény a **CHR$(0) + "K"** szöveget adja vissza, ha pedig a 🡆 nyilat, akkor a **CHR$(0) + "M"** szöveget. (Ennek az a speciális oka van, hogy csak.)

Dolgozzuk föl ezeket.

```
SELECT CASE a$
CASE CHR$(0) + "K": x = x - 1
CASE CHR$(0) + "M": x = x + 1
END SELECT
```

Ha az Olvasó most elindítja a programot, kellemetlen meglepetésben lesz része. Művünk **Illegal function call** üzenettel leáll az **elemrajz** 12. soránál, pontosabban a **LOCATE** utasításban, ami az elem kirajzolandó kockájának helyét beállítja. Az *F6* billentyűvel elérhető *Immediate* ablakban egy **PRINT** segítségével kideríthető, hogy **y** értékével van a baj, ez ugyanis nulla, márpedig a képernyőnek nincsen nulladik sora. De hogyan lett **y** nulla?

Morfondírozásunk egyelőre késedelmet szenved, előbb meg kell találni a hibát.

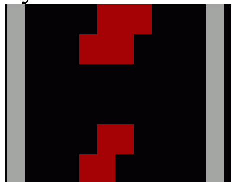
A programozó mindenekelőtt megnézi a képet, amit a programja csinált *(F4),* és lát rajta egy elemet.



Megnézi és összeráncolja szemöldökét. Nem ezt az elemet kérte. Azt mondta, **e = 1**, a *LISZTOJ* első betűje az **L,** ő ezt az elemet szeretné látni. Ez meg egy **S.** Illetve… **S?** Nem olyanok az arányai. A programozó megint belép az *Immediate* ablakba és beleírja:

```
elemrajz 3, 1, 6, 5
```

vagyis kérünk most nyomban egy **S** elemet kicsit lejjebb. Most ez látható a képernyőn:



Igen, ez tényleg nem **S** elem, nagyobb annál. Ez egy csomó kocka, amik vannak a képernyőn, nem egyetlen elem.

Ebből viszont nyomban levonhatunk egy következtetést. Az **elemrajz** rutin egyszeri meghívása egyetlen elemet rajzol a képernyőre. Ha a képernyőn nem egyetlen elemet látunk,

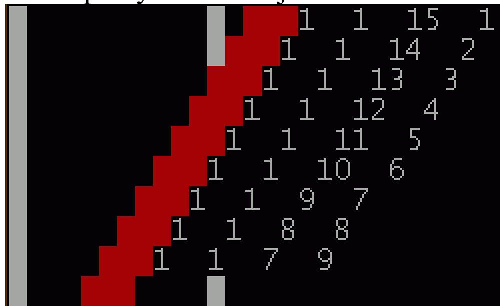
akkor az **elemrajz** legalább kétszer végrehajtódott. Éspedig úgy, hogy a paraméterek nem voltak ugyanazok, hiszen ha ugyanazok lettek volna, akkor kétszer (vagy többször) egymás után ugyanazt az elemet rajzolta volna ugyanoda, vagyis csak egyetlen elemet látnánk, **L**-t az első állásban.

Ha a paraméterek nem voltak ugyanazok, akkor mik voltak? Kérdezzük meg az **elemrajz**ot. Írjuk be a legelejére, vagyis 3. sornak: **PRINT e; i; x; y**.

A programozó megint lefuttatja a programot, megint hibaüzenet kap, *OK*-t nyom rá és megnyomja az *F4*-et. A képernyő két pontján lát számokat: közvetlenül a misztikus külsejű elem mellett az *1 1 7 1* számokat, az akna alatt pedig *1 1 6 2*-t olvas. Értelmezni kezdi a kémjelentést. Két számnégyest lát, tehát kétszer kellett végrehajtódnia az **elemrajz**nak. A két első szám azonos, tehát mindkét esetben **L**-et rajzolt az első állásban. A koordináták azonban mások. A felső számnégyes a 6, 2 koordinátákhoz helyezte az elemet, a másik a 7, 1-hez. Ez persze a kezdőkocka koordinátáit jelenti, márpedig az első állású **L** elemnél a kezdőkocka fölötti sorban is van kocka, a kezdőkockától eggyel jobbra és eggyel fölfelé. Ha a kezdőkocka pozíciója 7, 1, akkor ennek a kockának a pozíciója 8, 0. Nulladik sor nincs a képernyőn, nem csoda, hogy leáll. Éspedig akkor áll le, amikor a 7, 1-es koordinátákra rajzolandó elem felső kockáját kellene a képernyőn kívülre rajzolnia – a többi kockát már megrajzolta. Ebből ered tehát a különös kép: a kezdeti **L** elemünkre eggyel jobbra és fölfelé rárajzolódott egy újabb **L** elem, ami aztán félbemaradt.

Ez mind nagyszerű, no de miért akarja följebb és jobbra újrarajzolni? Miért pont följebb és jobbra, nem valami más irányban? Az újrarajzolás oka világos: a **mozgas** rutinban található **DO … LOOP** ciklus újra meg újra meghívja **elemrajz**ot és kirajzoltatja vele az elemet. De ugyanott kellene neki, hacsak le nem nyomjuk valamelyik nyílbillentyűt.

Miért mozdul el az elem a helyéről? Próbáljunk ki valamit. Eddig azt mondtuk a **mozgas**ban, hogy az elem az **x = 6: y = 2** helyre kerüljön. Amint elmozdult az elem fölfelé, kiment a képernyőről – tegyük át olyan helyre, ahonnan nem megy ki a képernyőről. Kerüljön ennek a sornak a végére **y = 10**, és nézzük meg így is.



A kép nagyon érdekes. Valamilyen erő hatására az elemünk lendületesen halad jobbra és fölfelé, amíg csak ki nem szalad a képernyőről. Bizonyára fúj a szél.

Valami megváltoztatja a koordinátákat, éspedig igen következetesen. A programozó nekiáll kideríteni, ki lehet az. Legjobb, ha folyamatosan figyeljük a koordinátákat, és amint megváltoznak, lecsapunk rájuk. Erre szolgál a *watch.* A programozó a képernyőre hívja a **mozgas** rutin szövegét és kiadja a *Debug* menü *Add Watch* parancsát, ahol

a *watch* szövegének ennyit ír be: **x**. Aztán újraindítja a programot a *Run|Restart* paranccsal, és az *F8*-cal végiglépeget az utasításokon. Amikor az **inic** és **aknarajz** sorokhoz ér a főprogramban, ezekre *F10*-et nyom, mert fölösleges a működésüket utasításonként végigkövetni, csak a **mozgas** kezdetétől érdekes, hogy mi történik. Amikor átlép az **x = 6** utasításon, a *watch* világoskék csíkjában a **mozgas x:0** felirat átvált 6-ra, **x** értéke tehát innentől 6. Nézzük tovább. Következik az **y = 2** utasítás, majd a **DO**, ezeket *F8*-cal végrehajtjuk, aztán jön az **elemrajz** hívása, ezt elég átlépni *F10*-zel…

Hoppá. Amint végrehajtottuk **elemrajz**ot, **x** értéke 7-re változott. Az **elemrajz** rutin változtatja meg **x** értékét!

A programozó olvasgatni kezdi **elemrajz** szövegét. Hát igen, csakugyan van benne egy **x = x + 1** utasítás, ami akkor hajtandó végre, ha a ceruzás kódszöveg **j** betűt tartalmaz. De mi köze a *kockák* helyét számon tartó **x**-nek az *elem* egészének helyét számon tartó **x**-hez?

Hát persze. A neve. **Elemrajz** rutinunk paraméterként megkapja az **x** és **y** változókat, aztán megváltoztatja az értéküket. A QB megengedi a rutinoknak, hogy megváltoztassák a paraméterként kapott változók értékeit, és a rutinból való visszatérés után a főprogrambeli változóban az új értéket találjuk. Persze csak ha a paraméter változó volt. Ha **elemrajz e, i, 6, 10**-et írtunk volna, az elem sose mozdult volna el, hiszen egy konstanst nem lehet megváltoztatni.

Végre tudjuk, hol a hiba, ki tudjuk javítani. Ha a **SUB** utasításban paraméterként megadott változókat nem változtatjuk meg a rutin belsejében, akkor minden rendben lesz. Legjobb, ha a **SUB** utasítást írjuk át, mert bent a rutinban többször szerepel **x** és **y**, többet kellene dolgozni, ha ott változtatnánk meg. Írjuk át a sort:

SUB elemrajz (e, i, ex, ey)

és közvetlenül alá kerüljön:

x = ex: y = ey

Így a rutinban **x** és **y** továbbra is ugyanazokat az értékeket képviseli, mint eddig, de a rutin paramétereként megadott **ex** és **ey** nem változik. Futtassuk le a programot.

Működik! Megjelenik az elem, és a program nem áll le! A kép kissé zavaros, mert a számok persze továbbra is ott vannak, és ha balra-jobbra mozgatjuk az elemet, akkor széles piros sáv lesz belőle, de a hibánkat kijavítottuk, az elem nem mozdul magától.

Állítsuk meg a programot *Ctrl-Break*kel, kapcsoljuk ki a *watch*ot *(Debug|Delete Watch),* és töröljük ki az **elemrajz**ból a **PRINT** utasítást. Újabb futtatás: az elem szépen mozog balra-jobbra, piros sávot húzva maga után. Ezt persze azért csinálja, mert kirajzolni kirajzoljuk, de aztán soha többé nem tüntetjük el, csak kirajzoljuk máshova is, ha elmozdul. Töröljük le a képről az elemet, ha elmozdult. Menjünk át a **mozgas** rutinba és a két **CASE** sor végén hívjuk meg az **aknarajz** rutint. Ez le fogja törölni.

A **mozgas** rutin még nincs kész, de nem árt pihenni egy kicsit. Idemásolom a program eddigi teljes szövegét ellenőrzés céljából, és ezt mostantól rendszeresen meg fogom tenni.

```
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, kfkljklk, kjkblkbk, kfkljklk
DATA kbkjlkjk, klkfjkfk, kbkjlkjk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: PRINT : NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
```

```
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
END SUB

SUB mozgas
e = 1: i = 1: x = 6: y = 2
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": x = x - 1: aknarajz
CASE CHR$(0) + "M": x = x + 1: aknarajz
END SELECT
LOOP
END SUB
```

Forgás

Mozgas rutinunk még tökéletesítésre szorul. Forgatni is kell az elemet.

A forgatás a 🡅 nyíllal történik, aminek a QB-ben **CHR$(0) + "H"** a kódja. A forgatás nem más, mint az irány, vagyis **i** megváltoztatása. Elemeinket úgy rendeztük el, hogy ha i-t növeljük, akkor az óramutató járása szerint elforgatott állásokat kapunk. Ha tehát el akarjuk forgatni az elemet az óramutató járása szerint, akkor **i**-t 1-gyel meg kell növelni. A negyedik állás után megint az első következik, vagyis ha **i** 4 volt, akkor 1-re kell változtatnunk.

Mindent tudunk, megírhatjuk a kódot.

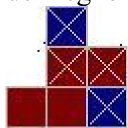
10 CASE CHR$(0) + "H": i = i + 1: IF i = 5 THEN i = 1

Ha az Olvasó beírja ezt a sort és kipróbálja, azt tapasztalja, hogy az elem elfordul ugyan, de rárajzolódik az előző állású elemre, s csak egy balra vagy jobbra mozdítás után jelenik meg helyesen. Csakugyan, elforgatás után megint az **elemrajz** hajtódik végre előbb, nem törlődik le az elem. Tegyünk be 11. sornak egy **aknarajz**ot, s így nézzük meg.

Így már jó. Kicsit csúnya ugyan, mert a kockák erősen téglalap alakúak, forgatás közben már nagyon látszik, hogy aránytalan az elem. Ezt nem különösebben korrigálnám, mert később úgyis áttérünk grafikus képernyőre, de a főprogramba beírhatunk a **CLS** elé egy **WIDTH 40** utasítást. Így szép négyzet alakú kockáink lesznek.

Érdemes átírni **e** értékét a **mozgas** rutin elején, kipróbálni a többi elem forgatását is. Ahogy próbálgatja az ember, az **S** elemnél nem egészen azt látja, amit kellene. Valahogy az az érzésünk, hogy nem azt kapjuk, amit kapni szeretnénk. Az elem kétségkívül elfordul 90 fokot, vízszintesből függőlegesbe, és újabb forgatásra továbbfordul vízszintesbe, de a programozónak úgy tűnik, hogy ennek nem így

kellene történnie. Ha az Olvasó is begépelte a programot, utánajárhat szavaimnak, de meg is mutatom rajzon, mit tapasztalok.

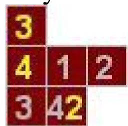


Forgatás előtt az **S** elem a pirossal jelölt kockákat foglalja el, forgatás után az × jelű kockákat, és ez a két állapot váltakozik. Csakhogy mi azt mondtuk, hogy az elemet a felső sor bal oldali kockája körül fogjuk forgatni, és annak a kockának a helyén (tessék ellenőrizni) az ×-elt állapotban az alsó sor jobb oldali kockája van. Hibásan van kódolva ez az állapot. Az **S** a harmadik elem, tehát a főprogram harmadik **DATA** sorának második szövege kell nekünk. Ez így szól: **"kfkljklk"**. Kocka, fel, kocka, le, jobbra, kocka, le, kocka. Való igaz. Ha a fekvő **S**-ben felül bal oldalt levő kocka körül forgatnánk, az most jobb oldalt lenne, és nem lehetne a szövegben **j** betű, csak **b**. Nekünk a **"klkfbkfk"** kód kell ide. Erre javítsuk ki a sor negyedik adatát is.

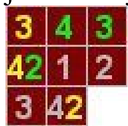
Elemünk még mindig nem úgy forog, ahogy kellene. Nincs más hátra, le kell rajzolni, hogy mit is szeretnénk, és alaposan áttekinteni a helyzetet. Először is legjobb, ha megszámozzuk a kockákat, végre alaposan kiismerjük köztük magunkat.



Így áll az elem az első állásban. Most forgassuk el az elemet az óramutató járása szerint 90 fokkal úgy, hogy az 1. kocka a helyén maradjon. Ez azt jelenti, hogy a 2. kocka az 1. alá kerül (oda, ahol most a 4. van), a 4. kocka az 1. mellé kerül balra, a 3. pedig a 4. fölé. Rajzoljuk rá az új helyzetet a régire:



Sárga számokkal jelöltem a második állás kockáit; az 1. kocka ugyanott marad, azt nem tüntettem föl még egyszer. Nézzük, mi lesz ebből az állásból, ha az 1. kockától kiindulva kódoljuk: **"klkfbkfk"**. Ez áll a harmadik **DATA** utasítás második helyén, ez tehát jó. Forgassuk tovább. 90 fok, óramutató szerint, az 1. kocka a helyén marad. A 2. kocka tehát oda kerül, ahol a sárga 3-as van; a 4. kocka az 1-es fölé; a 3. kocka pedig ettől jobbra. Rajzoljuk le:



Ó, hát persze. A zöld számokkal feltüntetett helyzet nem fog megegyezni a harmadik **DATA** sor harmadik kódjával, mert itt az 1. kocka az alsó sorban van, nekünk pedig soha nem jutott eszünkbe így kódolni. Pedig így kell. A zöld számokból a **"kbkjfkjk"** kód áll elő.

A rendszer most már kirajzolódik, különösebb gondolkodás nélkül elmondhatjuk, hogy az utolsó forgatásnál a jobb alsó sarokba is egy 3-as szám fog kerülni, a felső sorban a zöld 4-es mellé is kerül egy 2-es, és a jobb oldali 2-es mellé is kerül egy 4-es.



Valóban így is van, a piros számok a középső 1-essel szépen kirajzolják **S** elemünket. Az utolsó ceruzakód tehát **"kfkljklk"**.

A harmadik **DATA** sor tehát így alakul:

DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk

A programozó újra lefuttatja a programot és vidáman szemléli, amint az **S** elem most már csakugyan realisztikusan forog. Aztán eszébe jut, hogy három elem is van, aminek két-két állását egyszerűen a korábbiak lemásolásával készí-

tette el, és rosszkedvűen gondol az előtte álló feladatra. Nincs más hátra, mint elő-re, gondolja, csináljuk meg ezeket is.

A **Z** elem se forog helyesen, ez azonnal kiderül, amint a **mozgas**ba **e = 4**-et írunk és lefuttatjuk. Készítsük el a forgatási rajzot az előzőhöz hasonlóan.

4	23	4
23	1	23
4	32	4

Első állás (szürke): **"kbkjlkjk"**. Második állás (sárga): **"kfklbklk"**. Harmadik állás (zöld): **"kjkbfkbk"**. Negyedik állás (piros): **"klkfjkfk"**.

A harmadik kétállású elemünk az **I.** Nézzük meg ezt is, állítsuk **e**-t 2-re a **mozgas** elején. A programot lefuttatva azt látjuk, hogy ez nem annyira rossz, esetleg egyelőre maradhat. Ráérünk később bíbelődni vele, ha játék közben kiderül, hogy mégse az igazi.

```
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
WIDTH 40
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: PRINT : NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
```

```
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
END SUB

SUB mozgas
e = 4: i = 3: x = 6: y = 2
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": x = x - 1: aknarajz
CASE CHR$(0) + "M": x = x + 1: aknarajz
CASE CHR$(0) + "H": i = i + 1: IF i = 5 THEN i = 1
aknarajz
END SELECT
LOOP
END SUB
```

Esés

Elemeink most már jól mozognak és forognak, ideje, hogy elkezdjenek esni is. Egyelőre jobb lesz, ha nem maguktól esnek, hanem csak gombnyomásra, a 🡇 nyíl hatására. Így jobban megfigyelhetjük, mi történik.

Maga az esés nagyon egyszerű. Ha a 🡇 nyilat megnyomják, meg kell növelni **y**-t, és kész. A 🡇 billentyűre az **INKEY$** a **CHR$(0) + "P"** szöveget adja vissza, a **mozgas** rutinba tehát ez kerül:

12 CASE CHR$(0) + "P": y = y + 1: aknarajz

Elemeink innentől kezdve nagyszerűen esnek a 🡇 gomb megnyomására.

Most jön a neheze. Ha esik, akkor leesik. El fogja érni az akna fenekét, nekünk pedig ott meg kell állítani és az akna részévé tenni a továbbiakban. Vagyis meg kell csinálni az ütközést figyelő részt.

Morfondírozzunk. Az ütközés azt jelenti, hogy az elem egy (vagy több) kockája az aknában levő egy (vagy több) kockával azonos helyen van. Tehát nem egymáshoz érnek, hanem egész pontosan ugyanazt a helyet foglalják el. Ha az elem ütközik az akna tartalmával, akkor már nem szabad kirajzolni, hiszen a játékos

nem láthat olyat, hogy az elem kockája az akna kockájának helyére kerül. Ez azt jelenti, hogy az ütközés vizsgálatát a kirajzolás *előtt* kell elvégezni, és ha az elemünk ütközött, akkor nem is szabad kirajzolni. Sőt mást se szabad. Az elemünk ugyebár egy mozdítással került olyan helyzetbe, hogy ütközik az aknával. Ezt a mozdítást nem szabad megcsinálni, az elemnek azon a helyen és abban a helyzetben kell maradnia, ahol és ahogyan a mozdítás előtt volt. A mozdítás pedig a három irány és a forgatás közül bármelyik lehet. Vagyis az eseményeknek ebben a sorrendben kell történniük: ütközésvizsgálat, mozdítás, kirajzolás. Ehhez persze az ütközésvizsgálatnak úgy kell történnie, hogy a mozdítás *utáni* helyzetben vizsgáljuk az ütközést, nem abban a helyzetben, ahol az elem éppen akkor van.

Célszerű, ha a vizsgálatot egy külön programrész végzi, éspedig nem rutin, hanem függvény. Ez majd igaz vagy hamis értéket ad vissza a vizsgálat kimenetele szerint. Kétféleképpen is: az igaz jelentheti azt, hogy „igen, ütköztünk”, de azt is, hogy „igen, mehet a mozgás”. Tetszésünk szerint választhatunk. A második választásnak az az előnye, hogy azt írhatjuk: „ha mehet a mozgás, akkor tedd ezt és ezt”, vagyis **IF mehet(…) THEN** stb.

Mik lesznek a függvényünk argumentumai? A függvénynek azt kell megmondania, hogy az elemünk ütközik-e az aknával egy adott pozícióban. Kell tehát az elem sorszáma és állása, és kellenek a koordináták. Persze az új koordináták, amiket a mozgás után venne föl az elem. Vagyis a **mozgas** rutinban levő, az elemünket balra mozdító **x = x - 1** utasításból ez lesz:

IF mehet(e, i, x - 1, y) THEN x = x - 1

Csak akkor csökkentjük **x**-et, ha meggyőződtünk róla, hogy tehetjük, a csökkentett **x**-ű pozícióban nem ütközik az elem. Ez lesz a jobbra mozgatásnál a növelési utasításból is.

IF mehet(e, i, x + 1, y) THEN x = x + 1

A forgatással kicsit bonyolultabb a helyzet, mert nem mondhatjuk, hogy IF **mehet(e, i + 1…**, hisz lehet, hogy **i** már 4. Csináljuk azt, hogy előbb számoljuk ki a forgatás utáni **i** értékét anélkül, hogy magát **i**-t megváltoztatnánk, és ha mehet a dolog, akkor változtassuk meg **i**-t. Jelenleg így szól a programszöveg:

```
CASE CHR$(0) + "H": i = i + 1: IF i = 5 THEN i = 1
aknarajz
```

Legyen **ii** a forgatás utáni új irány:

```
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
```

és most nézzük meg, hogy mehet-e a dolog. Ha igen, tegyük át **i**-be az **ii**-ben beállított értéket.

```
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
```

A negyedik mozdítás, a lefelé történő ugyanúgy megy, mint az első kettő.

```
IF mehet(e, i, x, y + 1) THEN y = y + 1
```

Eddig megvolnánk, írjuk meg a **mehet** függvényt. Odáig minden világos, hogy *Edit|New FUNCTION,* a név **mehet**, és az ablakba beírjuk az argumentumokat. Ez lesz belőle:

```
DEFINT A-Z
FUNCTION mehet (e, i, x, y)
END FUNCTION
```

Morfondírozzunk tovább. Hogyan tudjuk meg, hogy az elemünk a megadott pozícióban ütközik-e az akna tartalmával? Úgy, hogy megnézzük mind a négy kockáról, hogy azon a helyen, ahova kerülni fog, mi van az aknában. Honnan tudjuk, hogy hova kerülnek az elem egyes kockái? Onnan, hogy végigcsináljuk ugyanazt, amit az elem kirajzolásakor, csak rajzolás helyett ellenőrzünk. Ennek legjobb módja, ha átmegyünk az **elemrajz**ba, kijelölünk mindent a **SUB** és az **END SUB** között, és áthozzuk ide.

```
DEFINT A-Z
FUNCTION mehet (e, i, x, y)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END FUNCTION
```

Az első sor rögtön emlékeztet minket arra a problémára, ami korábban oly sok fejtörést okozott az elsétáló elemmel. Itt is megváltoztatjuk **x** és **y** értékét, ezért nem szabad, hogy ők argumentumok legyenek. Írjunk csak itt is **ex**-et és **ey**-t formális argumentumnak.

A két **COLOR** utasítás természetesen nem kell, hiszen ez a függvény nem változtat a képernyő tartalmán, tehát színezgetnie se kell. És nem kell a **LOCATE** és a **PRINT** sem, a **CASE "k"** után egészen más dolog fog állni. Amikor az itt levő utasítások végrehajtódnak, az **x** és **y** változók arra a helyre mutatnak, ahol az elemünk soron levő kockája található. Vagyis itt kell ellenőrizni. Meg kell néznünk, hogy ezen a helyen mi van az aknában. Ha 1, akkor ütköztünk. Ha ütköztünk, akkor a mozdítás nem mehet, vagyis a **mehet** függvénynek nullát kell visszaadnia. A függvény visszaadott értékét úgy állítjuk be, hogy a függvény nevének mint változónévnek adunk értéket.

```
CASE "k": IF akna(x, y) THEN mehet = 0
```

Ez csak akkor ér valamit, ha van olyan eset is, amikor más értéket adunk vissza. Mikor adunk vissza más értéket? Ha mehet a mozdítás. Ez akkor történik, ha nem ütközünk. A **FUNCTION**-nel definiált függvények alapértelmezésként mindig

nullát adnak vissza, vagyis egy **mehet = 0** utasítás csak akkor csinál változtatást, ha előzőleg nem 0-ra volt beállítva a visszaadandó érték. Hát állítsuk másra. Tegyük a függvény legelejére, mondjuk a 3. sor végére a **mehet = 1** utasítást. Innentől tehát a **FOR** ciklusunk végrehajtása úgy kezdődik, hogy **mehet** értéke már 1, és ha ütközést tapasztalunk, akkor állítjuk 0-ra. Ezután viszont már soha nem állítjuk vissza 1-re, vagyis ha a négy kocka közül csak egy is ütközött, akkor **mehet** 0-t fog visszaadni. Készen vagyunk.

Azazhogy mégse vagyunk egészen készen, mert a program futtatásakor az derül ki, hogy bal oldalt és jobb oldalt csakugyan nem hajlandó rávinni az elemet az akna falára, szépen hozzásimulva megtorpan; de lent ez már nem így van, az elem belenyomódik az akna fenekébe, sőt túl is megy rajta. Vajon miért?

Némi morfondírozás kell, hogy kigondoljuk, hol kell keresni a hibát. Lehet, hogy az ellenőrzés után van a hiba: a lefelé mozgásnál rosszul írtunk valamit, ellenőriz és megállapítja, hogy ott ütközés van, de ennek ellenére engedi lefelé vinni az elemet. Lehet, hogy magában az ellenőrzésben, bizonyos esetekben nem veszi észre, hogy ütköztünk. Vagy lehet, hogy mi csak azt hisszük, hogy az akna feneke „aknafenékből van”, mert ezt rajzolja ki, de igazából az **akna** tömb nullákat tartalmaz azon a helyen? Ez utóbbi mondjuk nem túl valószínű, mert akkor miért jelennének meg kockák, ugye.

Nézzük az első esetet: vajon a lefelé mozgás ellenőrzése után engedjük-e a mozgást akkor is, ha ütköztünk? A **mozgas** rutinban ez áll:

```
CASE CHR$(0) + "P": IF mehet(e, i, x, y + 1) THEN y = y + 1: aknarajz
```

Hát ez elég egyértelmű. Csak akkor növeljük **y**-t, ha a **mehet** zöld utat adott neki, sehol máshol nem szerepel **y** növelése, tehát nem ez a probléma.

Azt már nem tudjuk ilyen könnyen, csak a programszöveg elolvasásával megnézni, hogy a **mehet** észreveszi-e az akna fenekét. Kémkednünk kell egy kicsit. Csináljuk azt, hogy a **mehet** függvény jelezze a képernyőn, hogy a vizsgálat során talált-e ütközést vagy sem. Menjünk be az ablakába és közöljük vele, hogy amikor ütközést talál, tegyen egy jelet a képernyőre. Legjobb, ha az aknán kívülre teszi, mert akkor nem törli le az **aknarajz**.

```
CASE "k": IF akna(x, y) THEN mehet = 0: LOCATE 10, 20: PRINT "*"
```

Próbáljuk ki. Semmi különös nem történik, amíg az elemet neki nem nyomjuk az akna falának; ekkor a képernyő közepén megjelenik egy csillag, ami a további mozgatások hatására elindul fölfelé, és csakhamar távozik a képernyőről. Hogy miért, azt most ne firtassuk, elég annyi, hogy így legalább nem kell letörölni. Vigyük hát le az elemet az akna fenekére és nézzük meg, kapunk-e csillagot.



Nem kaptunk csillagot mindaddig, amíg a képen látható helyzetben meg nem nyomtuk a 🡇 nyilat, csak hát ennek már két sorral följebb meg kellett volna történnie, hiszen a képen levő fehér csík az akna feneke. Az elem már ki is lóg az aknából.

Lehetséges volna, hogy az **aknarajz** följebb rajzolja az akna fenekét, mint ahogy mi elhelyeztük? Persze hogy lehetséges, épp csak nem valószínű. A progra-

mozó azonban összehúzza a szemöldökét. Programozónak lenni nem kevés szemfülességet is igényel. Már használtuk, amikor észrevettük, hogy a képernyőn nem **S** elem jelenik meg, hanem egy vastagabb valami, most pedig arra figyelünk fel segítségével, hogy az elem kilóg a képernyőről az akna feneke alatt, holott emlékeink szerint mi úgy méreteztük az aknát, hogy a képernyő egész magasságában végignyúljon. Nézzünk csak utána.

A főprogram tetején ott van az **aknay = 25** beállítás, ez konstans, ki van zárva, hogy bárki-bármi átállította volna később. Az **inic** rutinban ott van, hogy az **aknay**-odik sorba rajzoljuk az akna fenekét. A képernyő 25 soros. Akkor hogyan lehetséges, hogy az akna feneke alatt még van hely bármire?

A programozó hirtelen a homlokára csap, az összehúzott szemöldökök fölött. Persze! Mitől is szaladgál olyan lelkesen az a csillag fölfelé? Mert olyan utasítás nincsen a programban, hogy írj csillagot a tizedik sornál följebb. Kifejezetten a tizedik sorba lett mondva. Mi okozhatja, hogy valami, ami a képernyő egy adott pontján volt, följebb kerül? A scrollozás. A **PRINT** automatikusan följebb görgeti a képernyőt, ha az betelt, ezzel a csillag följebb kerül, éppenséggel az akna tartalma is, de az nem látszik, mert azt folyton újrarajzoljuk. Ezért van az a szörnyű villogás is, mindenféle kockák villannak föl a képernyőn, mert scrolloz, ezáltal a képernyő tartalma följebb kerül, aztán az elcsúsztatott aknára rárajzoljuk az elcsúsztatás nélküli aknát, ezzel eltüntetve a nyomokat saját magunk elől. Ezért nem jöttünk rá mostanáig semmire.

Mikor scrolloz a **PRINT** egészen pontosan? Amikor a képernyő legalsó sorába *Enter* karaktert kell kiírnia. Hol történik ilyesmi? Az **aknarajz**ban, természetesen, ezt gondolkodás nélkül megválaszolhatjuk. Ugyanis a csillagot kiíró utasítástól eltekintve két helyen van kiírás a programban, az **aknarajz**ban és az **elemrajz**ban, de utóbbi oda ír, ahol az elem éppen van, nem a képernyő utolsó sorába. És nem ír ki *Enter*eket, csak piros kockákat.

Nézzük meg az **aknarajz** szövegét.

```
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: PRINT : NEXT
```

Ott van, ni, az utolsó sorban. Egy paraméter nélküli **PRINT** utasítás megfelel egy *Enter* kiíratásának. Hát töröljük ki, gondoskodjunk máshogyan arról, hogy az előző sor **PRINT**-jei a megfelelő helyre írjanak. **LOCATE**-tel. Ezt célszerűen az **IF** elé fogjuk írni, paraméterei a koordinátáink, vigyázva a sorrendre: **LOCATE y, x**.

Futtassuk le a programot! Láss csodát, a villogás megszűnt, a csillag nem rohan fölfelé, az akna feneke szemmel láthatóan lejjebb került, és az elem szépen nekiütközik, nem nyomul bele. A problémát megoldottuk, az esés kész. A csillag kiírását ki is vehetjük.

```
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
WIDTH 40
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT
```

```
FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
e = 4: i = 3: x = 6: y = 2
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P": IF mehet(e, i, x, y + 1) THEN y = y + 1: aknarajz
END SELECT
LOOP
END SUB
```

Ragadás

Azazhogy bocsánat, az esést még nem egészen oldottuk meg. Arról is gondoskodni kell, hogy amikor az elemünk elérte az akna fenekét, akkor ott megkövüljön, az akna részévé váljon. Legyen a feladat neve **RAGADÁS,** és morfondírozzunk el róla.

A **RAGADÁS** feladatot akkor kell végrehajtani, ha a **mehet** függvény 0-t ad vissza. Azaz nem. A függvény akkor is 0-t ad vissza, ha azért nem mehet a mozgás, mert az elemet nekinyomtuk az akna valamelyik oldalának, de ott nem szabad megragadnia. Csak akkor ragadhat, amikor az akna fenekét éri el.

Honnan tudjuk, hogy ez mikor történt? Onnan, hogy amikor a **mozgas** rutin lefelé próbálja mozdítani az elemet (megnyomtuk a 🡇 nyilat), és a **mehet** függvény 0-t ad vissza, akkor a lefelé mozgás hiúsult meg, és az elemünknek meg kell ragadnia. Vagyis a **mozgas** rutinnak erre a részére van szükségünk:

```
CASE CHR$(0) + "P": IF mehet(e, i, x, y + 1) THEN y = y + 1: aknarajz
```

Ha nem mehet, akkor ragad. Szedjük szét a szöveget és tegyünk bele egy rutinhívást.

```
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas
END IF
```

Rendben, morfondírozzunk még tovább. Hogyan csináljuk a ragadást? Hát úgy, ahogy a kirajzolást, de ahelyett, hogy piros kockát tennénk a képernyőre, 1-est fogunk tenni az **akna** tömbbe. Csináljunk egy **ragadas** nevű rutint és másoljuk bele az **elemrajz** szövegét.

```
SUB ragadas (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB
```

A paramétereket már ki is javítottam **ex**-re és **ey**-ra, mielőtt baj lesz a dologból. A **COLOR** utasításokat persze most is kitöröljük, és a **LOCATE** és **PRINT** utasítást is. Kiírás helyett az aknába rakjuk a kockákat. Ez nagyon egyszerű lesz: **x** és **y** azokat a koordinátákat tartalmazza, ahova 1-est kell tennünk, tehát **CASE "k": akna(x, y) = 1** és kész. Vagyis csodát kész, nem indul el a program, hiszen a **mozgas** 15. sorában paraméter nélkül próbáltuk meghívni **ragadas**t. Ez pedig nem jó ötlet, mert át kell neki adni az elem adatait. Tehát ide **ragadas e, i, x, y** kerül. Vagy **y + 1**? Nem, mert az már arra a helyre mutatna, ahol az elem beleütközik az aknába. Így futtassuk le.

A dolog tökéletes. Az elem úgy megragadt az akna fenekén, hogy onnantól kezdve nem történik semmi, akárhogy nyomkodjuk a nyilakat, a képernyő változatlan marad. Talán ilyen jó munkát mégse végezzünk.

Vajon miért nem mozdul? Azt senki se mondta neki, hogy ragadás után indítson el egy új elemet a képernyő tetejéről, tehát logikus, hogy olyat nem kapunk. Az is logikus, hogy lefelé nem mozdul, mert nincs hova. Balra-jobbra mozgatni és forgatni pedig valószínűleg azért nem tudjuk, mert a **ragadas** rutin csakugyan 1-eseket helyezett az aknának arra a részére, amit az elemünk tölt meg, és akármerre

mozdítjuk az elemet, valamelyik kockája mindenképpen beleütközik az egyik 1-esbe (valamelyik szomszéd kocka megkövült képébe), tehát a program megtagadja a mozdítást. Legalábbis az a feltevésünk, hogy így van.

Ellenőrizzük. Tegyünk a **mozgas** 15. sorának végére, a **ragadas** hívása után egy **y = 2** utasítást, ezzel visszaküldve az elemet a képernyő tetejére. Azzal most ne vacakoljunk, hogy másik elemet válasszunk helyette vagy más irányba forgassuk, lényeg az, hogy az elemünk olyan helyre kerüljön, ahol van mozgástere.

Ez az. Az elem ott piroslik az akna fenekén, megnyomjuk a 🡇 nyilat, és az elem megjelenik az akna tetején is. Megint megnyomjuk a 🡇 nyilat, és az elem lép egyet lefelé, ugyanakkor az akna alján fekvő példány pirosból szürkére vált. Mivelhogy ekkor rajzolja ki a program újra az aknánkat. Célszerű lesz a **ragadas** rutin végére beírni egy **aknarajz** utasítást, hogy miután az elem megragadt, kirajzoljuk ebben az állapotában is.

Programunk nagyot haladt előre: immár szép kis lépcsőket, emelvényeket rakhatunk ki **Z** elemekből.

```
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
WIDTH 40
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
```

```
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
e = 4: i = 3: x = 6: y = 2
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: y = 2
```

```
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = 1
END SELECT
NEXT
aknarajz
END SUB
```

Sorok

Korábbi feladatlistánkból már csak egy tétel maradt, a **SOROK.** Annak vizsgálata, hogy betöltöttünk-e egy vagy több teljes sort, és ha igen, azok törlése és a fölöttük levő kockák lejjebb hozatala.

Erre akkor van szükség, amikor egy elem megragadt az aknában, úgyhogy jó ötletnek tűnik a **ragadas** végéhez írni. De legyen külön rutin, ártani nem árthat, úgyhogy a **ragadas** utolsó utasításaként csak ennyit írjunk: **sorok**. Ennek a rutinnak nem kell átadnunk a koordinátákat, az egész aknát végigvizsgálja, tehát paraméter nélküli rutin lesz. Hozzuk is létre.

Morfondírozzuk ki, mi lesz a **sorok** dolga. Végignézi az aknát sorról sorra és megnézi, hogy van-e olyan sor, ami csupa kockából áll. Fusson egy ciklus végig a sorokon, és egy másik végig a sor oszlopain:

```
FOR y = 1 TO aknay - 1
FOR x = 2 TO aknax - 1
```

Így az akna falait kihagytuk a dologból. Hogyan tudjuk ellenőrizni, hogy csupa kockából áll-e? Megszámolhatjuk a kockákat és ellenőrizhetjük, hogy annyi volt-e, ahány kitölt egy teljes sort – de sokkal egyszerűbb, ha megfordítjuk a dolgot. Ha találunk egyetlen üres helyet, akkor a sor *nem* csupa kockából áll.

```
IF akna(x, y) = 0 THEN tele = 0
```

Tele azt fogja jelenteni, hogy tele van-e a sor kockával. Ahhoz, hogy ez működjön, tegyük be a **FOR x** elé 5. sornak: **tele = 1**. Így minden sort azzal a feltételezéssel kezdünk el vizsgálni, hogy tele van, és ha üres kockát találunk, akkor nyugtázzuk, hogy mégsincs tele. Itt, a **tele = 0** után érdemes betenni egy **EXIT FOR** utasítást is, hiszen a sor további részét már teljesen fölösleges végignézni; időt nyerünk, gyorsabb lesz a programfutás.

Ha az **x** ciklus befejeződött (**NEXT**), akkor meg kell néznünk, hogy tele volt-e a sor, és ha igen, le kell zúdítanunk.

```
8 IF tele THEN zuditas y
NEXT
```

A **zuditas** egy rutin lesz, amely paraméterként a törlendő sor számát kapja meg. Magát a sort fölösleges kiüríteni; bőven elég, ha rámásoljuk a fölötte levő sort, arra az afölötti sort s így tovább. Más teendőnk a **sorok** rutinban, ha minden igaz, nincsen, rátérhetünk a **zuditas**ra.

```
DEFINT A-Z
SUB zuditas (y)
END SUB
```

Feladat: végigmenni az **y**. soron és rámásolni a fölötte levő, vagyis **y - 1**. sor tartalmát. Aztán végigmenni az **y - 1**. soron és rámásolni az afölötti sor tartalmát. És így tovább, amíg a legfelső sort is rá nem másoltuk az alatta levőre, a legfelsőt pedig ürítsük ki.

```
FOR yy = y TO 2 STEP - 1
```

Yy lesz a sor száma, amire éppen rámásolunk.

```
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
```

Végigmegyünk a sor minden kockáján (kivéve a falakat), és minden kockába belemásoljuk a fölötte levőt. Ez kész. Még töltsük fel nullákkal a legfelső sort:

```
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
```

Tegyünk a végére egy **aknarajz**ot, hogy az eredmény megszemlélhető legyen, és nézzük meg, mit csináltunk. Ehhez persze meg kell tölteni az akna legalább egy sorát. Ez úgy a legkönnyebb, ha függőlegesre állítjuk a **Z** elemeket és egymás mellé sorakoztatjuk őket.

Programunk működik. A tetrist szinte már megírtuk.

```
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
WIDTH 40
CLS
inic
aknarajz
```

mozgas

```
DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
```

```
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
e = 4: i = 3: x = 6: y = 2
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: y = 2
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = 1
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
```

```
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Új elem

Persze némi finomítás azért még kell. Mindenekelőtt az, hogy ne örökké **Z** elemekkel kelljen tetriseznünk, hanem rendesen váltogassák egymást. Csináljunk egy rutint, ami véletlenszerűen elemet választ. Legyen a neve **ujelem**, és legyen az a feladata, hogy a **mozgas**ban használatos **e**, **i**, **x** és **y** változókat beállítsa. Vagyis kapja meg őket paraméterül. Most ki fogjuk használni azt a tulajdonságot, ami korábban kellemetlen fejtörést okozott a magától elinduló elemmel: hogy a rutinok megváltoztathatják a paraméterül kapott változók értékét.

A rutint a **mozgas**ból fogjuk hívni, a 15. sorból, ahol a **ragadas** rutin hívása van; utána ne **y = 2** álljon, hanem **ujelem e, i, x, y**. És tegyük ugyanezt a 3. sorbeli értékadások helyére is, hogy már az első elem is véletlenszerű lehessen.

Maga **ujelem** egyszerű dolog lesz. Véletlenszerűen kisorsolja **e** értékét a lehetséges hétféléből *(LISZTOJ),* és ha már lúd, legyen kövér alapon **i**-nek is véletlenszerű értéket ad a lehetséges négyféléből. **X**-et beállítja úgy, hogy az elem az akna közepéről induljon, **y**-t pedig 2-re állítja, hogy az akna tetején legyen. (1-re nem célszerű, mert számos elemnek van kockája a kezdő kocka fölött is.)

```
DEFINT A-Z
SUB ujelem (e, i, x, y)
e = INT(RND * 7 + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB
```

Tegyünk egy **RANDOMIZE TIMER** utasítást az **inic**be, hogy az **RND** hatékonyan működhessen.

Tetrisünk most már úgyszólván tökéletes, mindent tud, amit egy egyszerű, de funkcionáló tetrisnek tudnia kell. Az elemek szépen forognak, esnek, elhelyezkednek az aknában, a betöltött sorok eltűnnek. Még két dolog hiányzik: az, hogy az elem magától is essen, a ↓ nyíl megnyomása nélkül, és az, hogy ha az akna megtelik, érjen véget a játék. Ezekkel azonban most még nem foglalkoznék, egyszerű okból. Ha ennyit akartunk volna csinálni, akkor most már csakugyan nem lenne más dolgunk, mint ezeket is beleírni, de hát még sehol se tartunk. A program továbbfejlesztése közben úgyis ki kell majd kapcsolni az elemek automatikus esését, az akna pedig úgyis ritkán telik majd meg, mert nem játszunk vele egyhuzamban addig. Úgyhogy ezek a dolgok még ráérnek.

```
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
WIDTH 40
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
IF akna(x, y) = 1 THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR 4
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
```

```
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = 1
END SELECT
```

```
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * 7 + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Színezés

Kezdjük továbbfejleszteni a programot. Az első, ami nyilván feltűnik az Olvasónak játszadozás közben, a program monoton látványvilága. A piros elemekből egy nagy, egyhangú szürke tömb lesz. Nem szép. Csináljuk meg színesre.

A tetrisek körében kétféle színezési rendszer ismeretes. A programok jókora része véletlenszerűen választ színt, amikor egy elem megjelenik a képernyőn, más részük viszont minden elemhez *(LISZTOJ)* egy-egy fix színt rendel hozzá, vagyis az elem színéből lehet tudni a formáját és megfordítva. Próbáljuk ki ez utóbbit.

A karakteres képernyőn tizenhat szín közül válogathatunk:

0		fekete
1		kék
2		zöld
3		cián
4		piros
5		lila
6		barna
7		világosszürke
8		sötétszürke
9		világoskék
10		világoszöld
11		világoscián
12		világospiros
13		világoslila
14		sárga
15		fehér

Válasszunk mindegyik elemnek egy színt. Például így:

L – lila (5),

I – cián (3),

S – sárga (14),

Z – zöld (2),

T – piros (4),

O – barna (6),

J – kék (1).

Tegyük ezeket bele a program elején egy tömbbe. Legjobb, ha **DATA** utasításokból olvassuk be őket, éspedig hozzárendelve az elemhez. Így az elem és a színe közötti összefüggés világos lesz. Írjuk tehát a színek kódszámait a **DATA** sorok végére, és tegyünk be egy új tömböt számukra:

DIM SHARED szin(1 TO elemek)

Az **inic** rutin 14. sorában pedig a két **NEXT** közé tegyük be: **READ szin(e)**. Máris betöltöttük színeinket.

Értesítenünk kell az **elemrajz**ot, hogy az elemeket ezentúl nem egységes pirosban, hanem az elem színével kell megjeleníteni. Ez nagyon könnyű lesz, hiszen az elem *LISZTOJ*-beli kódszámát **e**-ben a rutin már megkapta, a hozzá tartozó szín megvan **szin(e)**-ben, tehát csak annyi a teendőnk, hogy a **COLOR 4** utasítást lecseréljük **COLOR szin(e)**-re. Nézzük meg az eredményt.

Elemeink most már szép színesek, amíg a levegőben vannak, csak aztán szürkévé kövülnek meg. Nem csoda, hiszen az **aknarajz** semmit se tud az egészről.

Honnan fogjuk tudni, hogy az akna megkövült tartalmából melyik kockát milyen színnel kell kirajzolni? Logikus, hogy az **akna** tömbből, ahol ezentúl nem 0 jelzi az üreset és 1 a kockát, hanem 0 az üreset és minden egyéb szám a kockát.

Mégpedig a szín vagy az elem kódjával. Vagyis a sárga kockát vagy 14 jelzi (mert ez a sárga szín kódja), vagy 3 (mert ez az **S** elem kódja, és ez az elem sárga). Célszerűbb, ha a 14-et választjuk, mert így a megkövült kockák már függetlenek az őket egykor odaszállító elemek típusától. Az elemekből ugyebár a **ragadas** rutin csinál megkövült aknatartalmat, ez pedig úgyszintén paraméterként kapja az elem sorszámát; annyi csak a dolgunk, hogy a rutin 12. sorában **akna(x, y) = 1** helyett azt írjuk: **akna(x, y) = szin(e)**. És máris színekkel tele aknát kaptunk.

Ha az Olvasó most elindítja a programot, érdekes helyzetet fog tapasztalni. Az elemek odaragadásuk pillanatában eltűnnek, többé nem láthatók – kivéve az **L** elemet –, de ott vannak, mert a további elemek beléjük ütköznek. Az, hogy nem az történik, amit vártunk, logikus, hiszen az **aknarajz**ot még nem értesítettük a változásról. Nézzünk is bele.

Az **IF** utasítással van a probléma, ami úgy kezdi: **IF akna(x, y) = 1**, márpedig **akna(x, y)** most már elég sokféle érték lehet, nemcsak 0 vagy 1. Tehát írjuk ehelyett azt, hogy **IF akna(x, y) THEN**, ami azt jelenti, hogy nem egyenlő 0-val. Így minden olyan esetben, amikor az aknában nem 0 van, kockát fogunk kiírni. Elé pedig írjuk be 6. sornak: **COLOR akna(x, y)**. Hiszen az **akna** tömbben most a színkódok vannak, amiket egyszerűen átadhatunk a **COLOR** utasításnak.

Az első dolog, amit a program elindításakor tapasztalunk, az, hogy az akna falai elkékültek. Csakugyan, hiszen ott még a korábbi 1-esek vannak, ezek azonban színkódként értelmezve kéket adnak. Nem baj, attól még elkezdhetünk játszani az immár gyönyörű, tarkabarka tetrisszel.

```
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 7
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
WIDTH 40
CLS
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
```

```
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
COLOR akna(x, y)
IF akna(x, y) THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR szin(e)
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
```

```
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = szin(e)
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * 7 + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
```

```
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Raktárbővítés

További bővítési lehetőségnek kínálkozik, hogy a *LISZTOJ* készlethez további elemeket adjunk. A *LISZTOJ* azokat az elemeket tartalmazza, amik pontosan négy kockából összeállíthatók (az Olvasó nem fog tudni pontosan négy kockából olyan elemet összeállítani, ami nincs meg a *LISZTOJ*-ban), de számlálhatatlanul sok tetris teszi hozzá ezekhez a négynél kevesebb vagy több kockából álló elemeket. Az ilyen tetrisek változatosabbak, érdekesebbek, mint a hagyományos *LISZTOJ*-tetrisek.

Kezdjük az egészen kicsi kockaszámú elemekkel. Egyetlen kockából álló elem csak egyféle van, az egységkocka: ▢. Legyen a neve **K** (mint kocka), a ceruzás kódja logikusan szintén csak egyetlen **"k"**, nincs ceruzamozgatás, semmi. Forgathatatlan, akárcsak az **O** – a róla szóló **DATA** sor tehát ennyiből áll: **DATA k, k, k, k**. Írjuk oda a főprogram végére. Persze kell a sor végére egy színkód is – legyen fehér, 15-ös.

Most már nyolc elemünk van, az **elemek** konstanst ki kell javítanunk 8-ra. Mi még a teendő? Elvileg semmi. Indítsuk el a programot.

Semmilyen hibaüzenetet nem kapunk, de gyanúsan sokáig nem jelenik meg a **K** elem. Nézzük csak meg, miért. Állítsuk meg a programot és vegyük szemügyre az **ujelem** rutint. Hát persze, azt írtuk, hogy **e = INT(RND * 7 + 1)**, megfeledkezve arról, hogy a 7-es szám helyett **elemek**et kellett volna írni. Javítsuk ki és indítsuk újra a programot.

Ezúttal már megjelenik az elemek sorában a **K** is.

Folytassuk a bővítést. Két kockából is csak egyféle elem rakható össze: ■■, ami állhat vízszintesen vagy függőlegesen. Legyen a bal oldali a kezdő kocka, akkor a vízszintes állapot kódja **"kjk"**, a függőleges állapoté pedig **"klk"**. Az **I** elemnél megfelelt, hogy csak kétféle állapotot különböztettünk meg, hátha megfelel itt is. Adjunk neki 11-es (világoscián) színt, és írjuk be mindezt a főprogram végére.

DATA kjk, klk, kjk, klk, 11

Az **elemek** konstanst javítsuk 9-re és indítsuk el a programot.

Az új elem (nevezzük **D**-nek a dominó szóból) jól működik.

Három kockából kétféle elemet hozhatunk létre. Az egyik az egyenes sor: ■■■, nevezzük **H** elemnek a *három* szóból. Szintén állhat vízszintesen és függőlegesen.

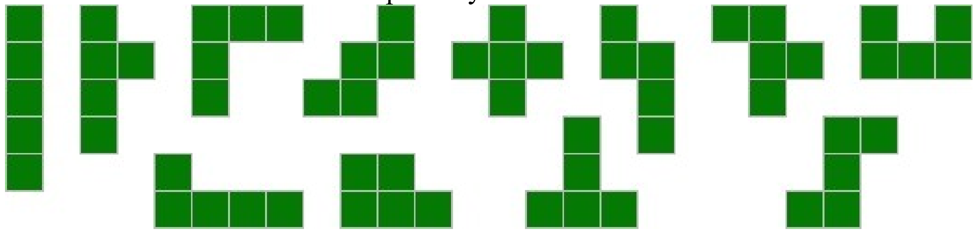
Itt persze a középső kockától kell indulnunk, így a vízszintes állapot kódja **"kbkjjk"**, a függőlegesé **"kfkllk"**, és a másik két állapot abszolúte tökéletesen megegyezik ezekkel, bármit csinálunk is. Csak a színt kell meghatározni: legyen világosszürke, 7-es. És megnövelni **elemek**et 10-re.

Ez az elem is nagyszerűen működik. Folytassuk!

A másik háromkockás elem az **L** kicsinyítése: , nevezzük **V**-nek. Ennek már négy állása van. A sarokkockával kell kezdenünk, a mutatott állás tehát **"kfkljk"**, 90 fokkal elforgatva **"kjkblk"**, továbbforgatva **"kbkjlk"**, végül pedig **"kbkjfk"**. A színe pedig legyen 13-as, világoslila. **Elemek = 11**, rajta!

Ez is tökéletes. Az 1, 2, 3 és 4 kockából álló elemek ezzel mind bekerültek tetrisünkbe.

Az ötkockás elemek csak a nehezebb tetrisekben használatosak; úgy gondolom, egyelőre kihagyhatjuk őket. Tizenketten vannak és némelyiküket elég nehéz beilleszteni az aknában létrehozott építményeinkbe.



A hat és több kockából álló elemeket pedig végképp jobb lesz kihagyni. Túl sokan vannak és túl bonyolultak.

```
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 11
CONST aknax = 12, aknay = 25
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
WIDTH 40
CLS
inic
aknarajz
mozgas
```

```
DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
LOCATE y, x
COLOR akna(x, y)
IF akna(x, y) THEN PRINT CHR$(219);ELSE PRINT " ";
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
COLOR szin(e)
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": LOCATE y, x: PRINT CHR$(219);
END SELECT
NEXT
COLOR 7
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
```

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = szin(e)
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB
```

```
SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Grafikusan

Munkánk igazán gyönyörű, senki sem vitathatja, de grafikus képernyőn még szebb volna. Akkor különböző mintákat adhatnánk a kockáknak, sok színből is válogathatnánk.

Ahhoz, hogy ezt megtehessük, választanunk kell egy grafikus üzemmódot. A játékprogramok ma már 800·600-as vagy még nagyobb felbontást használnak, nemritkán truecolorban. Erről mi nem is álmodhatunk, a QB megjelenésekor még nem tartott itt a technika. Viszont ismeri a 320·200-as, 256 színű üzemmódot, ami nagyon jól megfelel nekünk. A legjobb, ha az üzemmód bekapcsolásához szükséges **SCREEN 13** utasítást máris elhelyezzük az **inic** rutinban, a főprogramból pedig törölhetjük a mindjárt fölöslegessé váló **WIDTH 40** és **CLS** utasításokat.

De mielőtt fölöslegessé tesszük őket, számolnunk kell kicsit. Karakteres képernyőn egyszerű volt a dolog: egy karakterhely – egy kocka. Grafikus képernyőn ez már nem így megy: ki kell gondolnunk, mekkorák legyenek a kockák.

A függőleges méretből kell kiindulnunk, hiszen ez a kisebb, és ebben az irányban az akna meg is tölti a képernyőt, míg vízszintesen nem. Nézzük az ismert számokat. Van 200 képpontunk, és eddig volt 25 kockánk. 200 osztva 25-tel az 8. Vagyis 8 képpont magasak lehetnek a kockáink ahhoz, hogy ugyanannyi férjen belőlük a képernyőre, mint eddig. Mi lenne, ha fölkerekítenénk 10 képpontra? Akkor 20 kocka férne el a képernyőn, ez végül is még mindig elegendő a tetrisezéshez. Legyen hát inkább 10 képpont, akkor szebb rajzot tudunk nekik csinálni, és persze legyenek négyzet alakúak, vagyis 10·10 képpontból álljanak.

Legjobb, ha a 10-es számot máris tároljuk konstansként a főprogram elején: **CONST kocka = 10**, így később könnyen megváltoztathatjuk.

Gondolkodjunk tovább. Hol van programunkban olyan pont, ahol kockát kell kirajzolni a képernyőre? Nyilván ott, ahol a 219-es kódú karaktert kiíratjuk, ez a kocka. Ilyesmit két helyen is fogunk találni: az **aknarajz**ban, ahol az akna tartal-

mát ábrázoljuk vele, és az **elemrajz**ban, ahol az éppen eső elemet. Ha kockát két helyen is kell rajzolni, akkor célszerűbb lesz, ha írunk rá egy külön rutint, és azt hívogatjuk. Legyen a rutin neve **kockarajz**, és paraméterként kapja meg a koordinátákat meg a színt. Ennek a hívását tegyük az **aknarajz**ba ezek helyett a sorok helyett:

```
LOCATE y, x
COLOR akna(x, y)
IF akna(x, y) THEN PRINT CHR$(219); ELSE PRINT " ";
```

Legjobb, ha magára a **kockarajz** rutinra bízzuk a kockák kirajzolását és törlését is. Tehát csak annyi legyen itt:

```
kockarajz x, y, akna(x, y)
```

Itt először megadtuk a koordinátákat, aztán az aknából a színt. Ha ez nulla, akkor a rutin tudni fogja, hogy törölnie kell. Ámbár a fekete képernyőre nullás színnel, vagyis feketével kirajzolt kocka egyáltalán nem fog látszani, úgyhogy megfelel a törlésnek.

Nézzük az **elemrajz**ot. Innen a **LOCATE y, x: PRINT CHR$(219);** utasításokat töröljük, és mivel korábban egy **COLOR szin(e)** állt, ez kerül a helyükre:

```
kockarajz x, y, szin(e)
```

A **COLOR szin(e)** utasítást töröljük, és a rutin végén levő **COLOR 7** sem kell már.

Minden kész, megírhatjuk a rutinunkat.

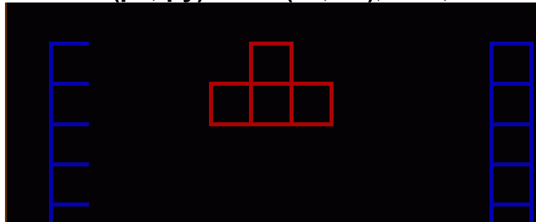
```
DEFINT A-Z
SUB kockarajz (x, y, szin)
END SUB
```

Morfondírozás következik. Megkaptuk **x**-et és **y**-t, de ezek az aknán belüli koordináták, az akna adott kockájára vonatkoznak. Rajzolni viszont csak grafikus koordinátákkal tudunk. Át kell alakítani egyiket a másikba. Egy kocka 10 képpontnak felel meg mindkét irányban (**kocka**), tehát ha ezzel beszorozzuk **x**-et és **y**-t, megkapjuk a koordinátákat. Legyenek ezek **px** és **py**, a *pont* szóból.

```
px = x * kocka: py = y * kocka
```

Ez lesz a kockánk bal felső sarka. A jobb alsó sarka nyilván 10 pontnyival jobbra és 10-zel lefelé lesz. Próbaképpen rajzoljunk egy egyszerű négyzetet a megadott színnel ezekre a koordinátákra:

```
LINE (px, py)-STEP(10, 10), szin, B
```



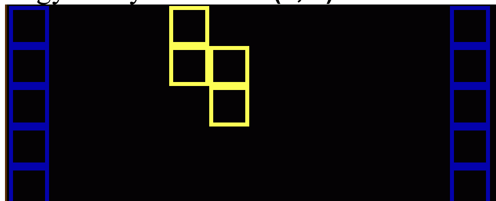
Nem az igazi. Először is a bal oldali falnak hiányzik a jobb szélső vonala, másodszor pedig az egész mindenség el van tolva egy kockányival jobbra és lefelé. A falak teteje fölött és a bal oldali faltól balra egy teljes kocka elfér még.

A bal felső sarokkocka koordinátái az aknában 1, 1. Csakugyan, itt a hiba. Ha az 1, 1 koordinátapárt 10-zel szorozzuk, akkor 10, 10-et kapunk, holott a képernyő bal felső sarka 0, 0-nál van. Vagyis először vonjunk le 1-et mindkét koordinátából, és csak azután szorozzuk be.

px = (x - 1) * kocka: py = (y - 1) * kocka

Eddig jó, a program indítása megmutatja, hogy az akna most már odailleszkedik a képernyő bal felső sarkához. Viszont bal oldalt még mindig csak fűrészfogak vannak. Vajon miért?

Nyilvánvaló, hogy a **LINE** utasítás **B**-s, vagyis téglalaprajzoló üzemmódja mindenképpen kirajzolja a téglalapot. Valaminek le kell azt törölnie onnan. A képernyőre azonban jelen pillanatban kizárólag a **kockarajz** rutin **LINE** utasításai rajzolnak, tehát nekik kell letörölniük. Egy adott kocka jobb szélét vélhetően a jobb oldali szomszédja tudja letörölni, ha rárajzolódik és nem mellé. Lehetséges, hogy ez történik? Ellenőrizzük. A bal felső kocka koordinátái az aknában 1, 1; ebből lesz az 1 levonásával és a 10-zel való szorzással 0, 0. Innen **STEP(10, 10)** a 10, 10-es koordinátájú pontba mutat, itt lesz a másik sarok. A kocka jobb oldali szomszédjának koordinátái az aknában 2, 1, a képernyőn (2 - 1) * 10 = 10, (1 - 1) * 10 = 0. Vagyis ennek a kockának az **x** koordinátája 10, miközben a bal oldali kocka jobb szélének szintén 10 az **x** koordinátája, egyszóval a két szomszédos kocka egymás felé néző szélét egymásra rajzoljuk. Miért is? Hát persze, azt mondtuk, hogy **STEP(10, 10)**, holott egy 0, 0-tól kiinduló 10·10-es négyzet nem 10, 10-nél ér véget, hanem 9, 9-nél. Mert az első pont a nullás, a második az egyes stb. Vagyis helyesen **STEP(9, 9)** kell a **LINE** utasításba.



Következtetéseink helyesek voltak, a kép most már jó – viszont a kockák érintkezésénél dupla vastagok a vonalak, ez főként az elemen látszik jól. Ez az, amivel viszont nem lehet mit csinálni. Majd olyan mintát fogunk tervezni, amin nem feltűnő a dolog. Egyelőre az is bőven elég, ha átírjuk a **LINE** utasítás végét **BF**-re, vagyis befestetjük a kockáinkat.

Van még egy probléma. Az aknának nincsen alja, és az elemek sorra eltűnnek a mélyben. Ó, persze, az akna magassága. Kiszámoltuk, hogy 200 képpont osztva 25 kockával az 8, hát akkor legyenek a kockák 10 képpont méretűek, legfeljebb egy kicsit kevesebb fér el a képernyőn – de elfelejtettünk szólni a programnak, hogy most már csak 20 kockányi mély az aknánk. Szóljunk neki, ott van az **aknay** konstans a főprogram elején, írjuk át 20-ra.

A játék remekül megy megint, a grafikus képernyőre való átalakítás hibátlan.

```
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
```

```
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 11
CONST aknax = 12, aknay = 20
CONST kocka = 10
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y)
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
```

```
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
```

```
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = szin(e)
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Hopp

Egy játékprogramban úgy lehet megtalálni a hibákat, hogy az ember játszik vele. Most is ez történt. A programozó egy függőlegesen álló **I** elemet közvetlenül az akna jobb falához illesztett és megnyomta a 🡅 nyilat. Az eredmény: **Subscript out of range**, mégpedig a **mehet** függvény **IF akna(x, y) THEN mehet = 0** utasításából.

Mivel az üzenet azt jelenti: a tömbindex kívül esik a megengedett határon, a programozó rögtön tudja, hogy csak az **akna(x, y)**-nal lehet a hiba. Megnyomja az *F6*-ot, hogy az *Immediate* ablakba kerüljön és beírja: **?x y**. A válasz a grafikus képernyő tetején jelenik meg: **13 13**. Tehát mindkét koordináta 13. A baj tehát az **x** koordinátával van, mert az csak 12-ig mehet.

A programozó újraindítja a programot és kísérletezik: mindegyik elemet oda-nyomja a jobb oldali falhoz és megpróbálja elfordítani. Többségük nem fordul el,

ha nincs rá módja, mások „ellökődnek” a faltól. Hibaüzenet kizárólag az **I** elemnél érkezik. Hát akkor nézzünk körül az **I** elem háza táján.

Elemünk a *LISZTOJ* szerint a második **DATA** sorban íródik le, ekként:

DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3

A hiba pedig akkor érkezik, amikor az elem függőlegesen áll és vízszintesbe akarjuk fordítani. Oké, akkor gondolkodjunk. Az aknánk 12 kocka széles (**aknax**), ebből az 1. és a 12. kocka a fal. Mielőtt el akarjuk fordítani az elemet, mindegyik kockája a 11. oszlopban van (hiszen függőlegesen áll). Ebben a pozícióban kell megvizsgálni, hogy a vízszintes állás szerint ütközik-e a fallal. A vízszintes állás ceruzakódja **"kbkjjkjk"**. Nézzük meg a **mehet** függvény lényegi részét.

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0
END SELECT
NEXT
```

Tehát először is vesszük az első karaktert, ez egy **k** lesz, ellenőrizzük **akna(x, y)**-t; ekkor **x = 11**. Aztán jön egy **b**, csökkentjük **x**-et 10-re, egy **k** miatt ellenőrizzük, aztán jön két **j**, kétszer megnöveljük, az 12, **k**, ellenőrzünk (12-nél falba ütköztünk, tehát **mehet**et 0-ra állítja a program), **j**, növelünk… hopp, akkor az már 13 lesz! És még van a szöveg utolsó **k** betűje, ami ellenőriztet velünk. Már az előző **k** betűnél beleütköztünk a falba, a **mehet = 0** érték eldöntetett, de még mindig ellenőrzünk, holott az elem túlszaladt a falon. Ezt meg lehet oldani: a **mehet = 0** után írjuk oda, hogy **EXIT FUNCTION**, és a végrehajtás majd szépen kilép az első ütközésnél.

Indítsuk el a programot és próbáljuk ki. A program most már nem áll le a kritikus esetnél – a hibát kijavítottuk.

```
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 11
CONST aknax = 12, aknay = 20
CONST kocka = 10
DIM SHARED akna(1 TO aknax, 1 TO aknay)
```

```
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y)
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
END SUB
```

```
SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = 0: EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = szin(e)
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
```

```
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Fennakadás

Valami azért még nem egészen az igazi. Ha odaviszünk egy **I** elemet függőlegesen a fal mellé és megpróbáljuk elfordítani, nem fordul el. Sőt. Ha a jobb oldali fal mellé úgy tesszük oda, hogy az elem és a fal között egy kockányi üres hely van, akkor sem fordul el. Bal oldalt ez nem így van, akkor elfordul. Nem az igazi. Szebb lenne, ha ilyenkor is elfordulna.

Morfondírozzunk. Amikor a függőlegesen álló **I** elem egy kockányi távolságban áll a jobb faltól, akkor a vízszintesbe forduló elem a következőképpen állna: . A karikával jelölt helyen van a kezdő kocka, hiszen úgy csináltuk meg, hogy a balról számított második kocka körül forduljon az elem. Az ×-szel jelölt kocka már egybeesik a fallal, ezért nem engedi ebbe az állapotba fordulni. Ha eggyel balra tolnánk, akkor már engedné, a játékos pedig semmi különöset nem venne észre, hiszen a kockák küllemre egyformák, mit zavarja őt az, hogy a második vagy a harmadik kocka körül fordul meg az eleme.

Hogyan csináljuk ezt? Ha a **mehet** függvény nullát ad vissza, de eggyel balra tolva már 1-et, akkor vigyük eggyel balra az elemet. Persze ugyanez bal oldalt is így néz ki, tehát ha eggyel jobbra tolva 1-et ad, akkor vigyük jobbra. De mit csináljunk, ha balra tolva meg jobbra tolva is 1-et ad, csak azon a szent helyen nem?

Morfondírozzunk tovább. Tegyük fel, hogy a **mehet** függvény figyeli, hogy a kezdő kockához képest balra vagy jobbra következett be az első ütközés. Hiszen tudja, hogy milyen koordinátákon helyezkedik el a kezdő kocka (megkapta argu-

mentumként), és tudja, hogy hányat lépked balra-jobbra. Meg tudja tehát mondani, hogy a kezdő kockától balra vagy jobbra ütközött-e az akna tartalmába. Ha balra ütközött, akkor mi majd jobbra toljuk a kockát; ha jobbra ütközött, akkor balra toljuk; ha mindkét irányban ütközött, akkor nem forgatjuk.

A **mehet** által visszaadott érték tehát már nemcsak kétféle lehet. Eddig így működött:

0 – ütközött;

1 – nem ütközött.

Most csináljuk így:

1 – nem ütközött;

2 – balra ütközött;

3 – középen ütközött;

4 – jobbra ütközött.

A **mozgas**ban pedig úgy használjuk föl ezt az értéket – persze csak a forgatásnál! –, hogy ha 1, akkor mozgatunk; ha 2, akkor megpróbálunk forgatás előtt jobbra mozogni, megnézve, hogy tudunk-e; ha 4, akkor ugyanígy balra. 3-nál nincs mit tenni, nem forgathatunk.

Kezdjük a **mehet** átírásával. Kapóra jön, hogy az argumentumban átadott oszlopkoordináta megmarad **ex**ben változatlanul, tehát a **mehet = 0** helyett csak anynyit kell tennünk, hogy ha **x < ex**, akkor **mehet = 2**, ha egyenlőek, akkor 3, ha **x** a nagyobb, akkor viszont 4. Ez viszont egy többágú **IF** (vagy **SELECT CASE**) utasítást igényelne, amit egy másik **IF**-be (az **IF akna(x, y)** kezdetűbe) kell betenni, és mindez egy **CASE**-ben van, márpedig a többsoros **IF** a QB szabályai szerint csak sor elején kezdődhet. Nagyon hosszú lenne. A programozó ilyenkor szívesebben trükközik.

Készítsünk egy kis táblázatot, ami megmondja, mire kell beállítani **mehet** értékét.

ha…	…akkor **mehet** értéke…
x < ex	2
x = ex	3
x > ex	4

No most. Tegyük fel, hogy **ex** értéke 10. Ha **x** értéke 9 vagy kevesebb (2-es eset), akkor **x - ex** negatív. Ha **x** értéke 10 (3-as eset), akkor **x - ex** nulla. Ha pedig **x** értéke 11 vagy több (4-es eset), akkor **x - ex** pozitív. Az előjelet a **SGN** függvény szempillantás alatt megadja nekünk, ha tehát azt írjuk, hogy **SGN(x - ex)**, akkor a 2-es, 3-as és 4-es esetekre rendre –1, 0 és 1 értékeket kapunk. Akkor nincs más dolgunk, mint hozzáadni 3-at, és megkapjuk a kívánt eredményeket. Ez a sor tehát így fog festeni:

```
CASE "k": IF akna(x, y) THEN mehet = SGN(x - ex) + 3: EXIT FUNCTION
```

Nagyszerű, a **mehet** kész is, lássuk a **mozgas**t. Itt először is ki kell javítani az összes **mehet**-hivatkozást, mert eddig csak annyit írtunk, hogy **IF mehet(…) THEN**, vagyis ha a visszaadott érték nem nulla, akkor mehet; most viszont egyál-

talán nem lesz nullás visszaadott értékünk, kifejezetten az 1 értéket kell vizsgálni. Vagyis három helyen a **mehet(…)** után írjuk be, hogy **= 1**; a forgatást pedig vegyük szemügyre tüzetesen.

```
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
IF mehet(e, ii, x, y) THEN i = ii: aknarajz
```

Az első sor kideríti, hogy melyik állásban kell lennie az elemünknek forgatás után (**ii**), a második pedig ellenőrzi, hogy abban az állásban mehet-e, és ha igen, átteszi **i**-be az új irányt.

Logikusan itt is be kell írnunk a **mehet**-hivatkozás után, hogy **= 1** (tegyük is meg, akkor le tudjuk ellenőrizni, hogy még mindig ugyanúgy működik-e minden a program futása során – igen, az elemek továbbra is rendesen viselkednek), viszont most van három más esetünk is, 2-3-4. Tehát alakítsuk ezt az **IF**-et **SELECT CASE**-zé, ami így kezdődik:

```
SELECT CASE mehet(e, ii, x, y)
CASE 1: i = ii: aknarajz
```

Oké. Mi történik, ha 2? Akkor bal oldalt ütköztünk, tehát jobbra kell mennünk. Vagyis idemásolhatjuk a jobbra mozgató programrészt, egy pici változtatással: nem **i**-vel, hanem **ii**-vel dolgozunk, mert *elforgatott állapotban* szeretnénk elmozdítani.

```
CASE 2: IF mehet(e, ii, x + 1, y) THEN x = x + 1: aknarajz
```

A 4-es eset a fordítottja ennek: jobbra ütköztünk, tehát balra kell menni.

```
CASE 4: IF mehet(e, ii, x - 1, y) THEN x = x - 1: aknarajz
```

A 3-as eset az, amikor a kezdő kockával azonos oszlopban ütköztünk. (Maga a kezdő kocka nem ütközhet egy forgatás eredményeképpen, hiszen olyankor sosem változtatja helyét, de a jobb oldali szomszédja forgatáskor alája kerül és ütközhet.) Ilyenkor nincs mit tenni, a forgatás nem hajtható végre, **CASE 3**-ra nincs szükség.

Jöhet az **END SELECT**, és próbáljuk ki művünket.

Ami azt illeti, nem jó. A függőlegesen álló **I** elem a jobb falhoz nyomva egyáltalán nem reagál, a bal falhoz nyomva viszont fordulási parancsra eggyel jobbra megy és *függőlegesen marad.* Nézzük meg azt is, mi történik, ha a faltól egy kockányi távolságban próbáljuk elfordítani. Bal oldalt ez nem probléma, eddig is elfordult, jobb oldalt viszont ugyanaz történik, mint bal oldalt a falhoz nyomva: távolabb megy a faltól, de nem fordul el. Persze, igaz is. Ha a **mehet** 2-t vagy 4-et ad vissza, akkor az elforgatott állapotot ellenőrizzük ugyan, de utána nem forgatunk. Tegyük be ebbe a két sorba a **THEN** után az **i = ii** utasítást is.

A bal falhoz nyomva elfordul vízszintesre. A jobb falhoz nyomva nem mozdul. A jobb faltól egy kockányira ismét vízszintesbe fordul. Ez már jobban hasonlít arra az állapotra, amit szerettünk volna elérni, csak zavaró, hogy a bal és a jobb fal nem azonosan hat az **I** elemre.

További morfondírozásra van szükség.

A két fal nyilvánvalóan azért nem hat egyformán, mert a vízszintesen fekvő **I** elem kezdő kockája balról a második, vagyis ha a fekvő elemet végével neki-

nyomjuk a bal falnak, akkor a fal és a kezdő kocka között egy kocka van: ; ha viszont a jobb falnak nyomjuk neki, akkor kettő: . Ahhoz, hogy ez utóbbi állapot álljon elő abból az állapotból, amikor az álló **I** neki van nyomva a jobb falnak, két kockával kellene balra vinni az elemet. Márpedig a **mozgas**ban elhelyezett új szövegrész legfeljebb eggyel viszi arrébb.

Kénytelenek leszünk tovább variálni. Nem elég, hogy a **mehet** visszaadja az irányt, hogy merre történt az ütközés: a távolságot is vissza kell adnia. (Még szerencse, hogy csak *x* irányban.) Persze a távolság sose lesz 2-nél nagyobb, sőt a 2-es érték is csak jobbra történő ütközésnél fordulhat elő, úgyhogy az eddigi értékek mellé fel tudjuk venni az 5-öst annak jelzésére, hogy jobbra két kockányival ütközött. (Mielőtt az Olvasó megkérdezi, mire ez a felhajtás, hiszen megcsinálhatnánk azt is, hogy az **I** elem negyedik állása ne legyen, hanem , elmondom, hogy az elemek bármelyik állásban odakerülhetnek a falhoz, és nagyon csúnyán nézne ki, ha az egyformán vízszintesen álló elem az egyik alkalommal így viselkedne, a másik alkalommal amúgy.)

Tehát nézzük a **mehet**et. Ez áll benne:

IF akna(x, y) THEN mehet = SGN(x - ex) + 3: EXIT FUNCTION

Ezt át kellene alakítani oly módon, hogy ha **x - ex = 2**, akkor 5-öt adjon vissza. Induljunk ki abból, amink van. Ez a szöveg most 4-et ad vissza ebben az esetben, hiszen **x - ex** pozitív, a szignuma 1, plusz 3 az négy. Csak hozzá kell adni egyet, ha **x - ex = 2**. Ha ezt a képletet kiíratjuk például **PRINT** utasítással, akkor –1-et vagy 0-t kapunk aszerint, hogy mennyi **x** és mennyi **ex**, hiszen az **=** jel a BASIC-ben műveleti jel, ami igaz vagy hamis értéket ad vissza két operandusának összehasonlítása után. Ha az **=** jel nem –1-et, hanem +1-et adna vissza az összehasonlítás igaz volta esetén, akkor csak hozzá kellene adnunk az **x - ex = 2** képlet értékét a már meglevő képletünkhöz, és kész lennénk; de mivel –1-et ad vissza, nem hozzáadjuk, hanem kivonjuk belőle. Negatív szám kivonása összeadást jelent. Az összehasonlítást persze zárójelbe tesszük, mert a BASIC hierarchiaszabályai szerint a kivonás előbb következik, mint az összehasonlítás. Tehát:

IF akna(x, y) THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION

Most menjünk át a **mozgas**ba és értelmezzük az 5-ös értéket. Legjobb, ha lemásoljuk a 4-es érték sorát (a **CASE 4** kezdetűt), kijavítjuk a 4-et 5-re, és az **x = x - 1** értékadást átírjuk **x = x - 2**-re. Kész a kocsi – próbáljuk ki.

A program még mindig nem azt csinálja, amit szeretnénk. A közvetlenül a jobb falhoz nyomott **I** elem nem fordul, az eggyel balrább léptetett elem pedig úgy fordul el, hogy a jobb vége (!) lesz abban az oszlopban, ahol addig az elem állt.

Debugoljunk. (Nem függ össze a BASIC történetével, de érdekes dolog, hogy miért lett a számítástechnikai hibakeresés szakszava a *debugging,* magyarul debugolás. Az egyik legelső elektronikus számítógéppel, az *ENIAC*-kal történt volt, hogy egy nap nem működött. A mérnökök nekiálltak a keresésnek, és meg is találták a hibát: a számtalan elektroncső egyikének csatlakozásánál egy lepke szorult a két vezeték közé, ott elégett és rövidzárlatot okozott. Eltávolították a tetemet, s a

nagykönyvbe beírták, hogy eltávolították a *bug*ot, vagyis bogarat. Ettől kezdve a hardver-, majd a programhiba neve bug lett, eltávolítása pedig debugging.)

Nézzük meg próbaképpen, hogy milyen értéket ad vissza a **mehet** függvény. Az **EXIT FUNCTION** előtti utasítást másoljuk oda még egyszer, de a **mehet =** részt cseréljük ki **PRINT**-re. Jelenjen meg a képernyőn, hogy milyen értéket adott.

Érdekes: amikor egy **I** elemet függőlegesen állítva a jobb falhoz illesztünk és megpróbálunk elfordítani, két szám jelenik meg egymás alatt, 4 és 5. Vajon miért?

Próbáljuk nyomon követni az eseményeket onnantól kezdve, hogy az elem a jobb falnál van és megnyomtuk a ↑ gombot. Legjobb, ha a **mozgas** rutinban közvetlenül a **CASE CHR$(0) + "H"** után beírunk egy **STOP**-ot, az **ujelem**et pedig átmenetileg átállíthatjuk úgy, hogy kizárólag függőlegesen álló **I** elemeket adjon, így gyorsabb a tesztelés:

```
e = 2'INT(RND * elemek + 1)
i = 2'INT(RND * 4 + 1)
```

A sorok eredeti tartalmát **'** jellel kiiktattuk, így könnyű lesz őket visszaállítani.

Vigyük a falhoz az elemet és nyomjuk meg a ↑ gombot. A végrehajtás azonnal megszakad a **STOP** utasításnál. Innentől az *F8* és *F10* billentyűkkel követhetjük a program útját.

Először a **mehet** hívódik meg. A **PRINT** egy 4-est ír ki, mielőtt kilépünk a függvényből. A 4-es a kezdő kocka jobb oldali szomszédjának ütközését jelenti; ez logikus, hiszen az elemet ilyen helyzetbe akartuk fordítani (az × a fallal egybeeső kockát jelenti). Menjünk tovább *F8*-cal.

A **mozgas** rutin **CASE 4** kezdetű sora ismét végrehajtatja **mehet**et, és most 5-öt ír ki kilépés előtt. Írassuk ki (*Immediate* ablak) **ex** értékét. 10. Vagyis ezúttal a függvény úgy hívódott meg, hogy a kezdő kocka a 10. oszlopban volt, a fal pedig ugye a 12. oszlop, tehát ebben a helyzetben volt az elemünk. Tagadhatatlan, hogy ebben a helyzetben a kezdő kockától kettővel jobbra következik be az ütközés, vagyis az 5-ös szám kiírása jogos. Ha a **mehet** második meghívása előtt nem eggyel, hanem kettővel vittük volna balra az elemet, akkor most nem lenne ütközés.

Próbáljunk ki valamit. Írjuk át ezt a sort:

```
CASE 4: IF mehet(e, ii, x - 1, y) = 1 THEN i = ii: x = x - 1: aknarajz
```

erre:

```
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN i = ii: x = x - 2: aknarajz
END IF
```

Vagyis ha forgatásnál jobbra ütköztünk, és az eggyel balra vitt elem még mindig jobbra ütközött, akkor próbáljuk meg kettővel balra vinni, hátha akkor már nem ütközik.

Forgatáskor a program persze megáll a **STOP** miatt, de ha továbbküldjük *F5*-tel, az elem megjelenik vízszintesen a fal mellett. Vegyük ki a **STOP**-ot és a **mehet**ből a **PRINT**-et, próbáljuk ki így is. Programunk végre az elvárt módon működik. Az **ujelem**ben is állítsuk vissza a két kiiktatott utasítást!

```
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z
CONST elemek = 11
CONST aknax = 12, aknay = 20
CONST kocka = 10
DIM SHARED akna(1 TO aknax, 1 TO aknay)
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y)
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
```

```
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y) = 1
akna(aknax, y) = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay) = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y) THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) = 1 THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) = 1 THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
i = ii: x = x - 1: aknarajz
```

```
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN i = ii: x = x - 2: aknarajz
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y) = szin(e)
END SELECT
NEXT
aknarajz
sorok
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y) = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy) = akna(x, yy - 1)
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1) = 0
NEXT
aknarajz
END SUB
```

Játszadozás

Miután két fejezetet hibajavítással töltöttünk, ideje visszatérni a bővítéshez. Próbáljunk ki új dolgokat, játszadozzunk a programmal. Mi történik például, ha megváltoztatjuk az akna méretét? A magassága kötött (hacsak nem kicsinyítjük le a kockákat), de a képernyő széltében még rengeteg hely van. 12 oszlopot használunk, pedig a 320 képpontra 32 oszlop is kiférne. Próbáljuk meg **aknax**-et 32-re állítani és így is játszani vele!



Így is minden tökéletesen működik, az elemek az akna egész óriási szélességében mozgathatók, a betöltött sorok eltűnnek. Egy baj van: elvettük a játék izgalmát. Eddig se volt sok neki, mert a tetrisnek az a lényege, hogy egyre gyorsabban esnek az elemek, egyre nehezebb elhelyezni őket, de ezt magyarázhattuk azzal, hogy ez még egy munkaverzió, ahol az elemek szándékosan nem esnek maguktól, pláne nem egyre gyorsabban. Most viszont mi sem könnyebb annál, mint helyet találni az elemeknek, a hatalmas aknában rengeteg hely van; az építmény csak akkor lesz magasabb, ha a játékos szándékosan nem tölti be a sorokat.

De tegyük fel, hogy a játékos valamilyen módon motiválva van arra, hogy magasabb építményt hozzon létre. Ez érdekesebb lenne, mint egyszerűen visszaállítani **aknax**-et 12-re. (Ámbár érdemes azt is kipróbálni, hogy milyen lesz a játék, ha még ennél is kisebb értékre állítjuk. A legkisebb játékra alkalmas érték 6, hiszen két kocka kell a falaknak és négy az aknának, hogy a fekvő **I** elem is elférjen.) A széles akna nem szokása a tetriseknek – most csinálhatunk valami eredetit.

Mi motiválhatná a játékost arra, hogy magasabbra építkezzen? Például feladatául tűzhetnénk ki, hogy a tetriselemekből megépítsen valamilyen alakzatot. Ha sikerül, újabb alakzattal jutalmazzuk.

Ezen azért morfondírozni kell egy kicsikét. Van egy felépítendő alakzat (mondjuk egy 5 kocka szélességű függőleges torony az akna aljától 15 kocka magasságban), amit berajzolunk az aknába, hogy a játékos lássa, mit kell felépítenie. Persze ez a rajz nem akadályozza az elemek mozgását. Nem tiltjuk meg, hogy elemeket tegyen az előírt alakzaton kívüli területre, csak azok persze nem számítanak bele az alakzatba. Ha megtölt sorokat, akkor az alakzaton belüli és kívüli kockák egyformán lejjebb kerülnek, ami vagy hasznos a játékos építkezése szempontjából, vagy nem, a helyzettől függ.

Hogyan jelöljük az alakzatot? Például valamilyen sötétebb mintával, amit nem lehet összetéveszteni a tetriselemekkel. Legjobb, ha a minta ott is látszik, ahol kocka van, így tudjuk, hogy az a kocka beleszámít az alakzatunkba.

Hogyan *tároljuk* az alakzatot? Az **akna** tömbben lesz a helye, ez egészen nyilvánvaló. A tömb jelenleg a négyzetrács minden kockájáról egy számértéket tárol, ami 0, ha nincsen ott kocka, és a kocka színe, ha van. Ezt jobb lesz nem bolygatni. Inkább bővítsük ki a tömböt egy mezővel, ami az alakzattal fog foglalkozni.

Ehhez először is rekorddá kell alakítanunk az **akna** tömböt. Kell egy rekordtípus, legyen például a neve **aknahely**, mert az akna egy-egy koordinátahelyének adatait tartalmazza. A tömbben tárolt jelenlegi értéknek adjuk a **szin** nevet:

```
13 TYPE aknahely
szin AS INTEGER
END TYPE
```

Ez kerüljön a főprogramba, a tömb deklarálását pedig így írjuk át:

```
DIM akna(1 TO aknax, 1 TO aknay) AS aknahely
```

Több mezőt egyelőre fölösleges csinálni – mielőtt használhatnánk őket, úgyis át kell írnunk az egész programban minden egyes hivatkozást az **akna** tömbre. Erre jó módszer, hogy elindítjuk a programot, azonnal meg fog torpanni egy hivatkozásnál.

A **zuditas** rutin **akna(x, 1) = 0** sorában állt meg **Type mismatch** üzenettel. Ez az utasítás ugyanis számot próbál értékül adni **akna** egy elemének, ami viszont most már rekord, nem numerikus változó. Írjuk át: **akna(x, 1).szin = 0**. És végig a programban mindenütt.

Az elindítom–megtorpan módszer mégsem tökéletes, mert a **zuditas**ban van olyan sor is, ami szintaktikusan helyes marad így is: **akna(x, yy) = akna(x, yy - 1)**. Ezt is írjuk át, és nézzük végig a programot továbbiak után. (Nem lesz több.)

Eddig jó, programunkban átvezettük a változást – jöhet a bővítés.

Először is kell egy mező, ami az alakzatra vonatkozó információkat tartalmazza. Ez is lehet egy egész változó, ami 1, ha az alakzat részét képezi az a hely, vagy 0, ha nem. Legyen a neve **tolt**, mert az alakzatot ki kell tölteni.

```
15 tolt AS INTEGER
```

Most látogassunk el **inic**be, és készítsünk egy alakzatot. Az elején már van egy **FOR y = 1 TO aknay - 1** ciklus, ez nekünk jó is lesz, csak írjuk bele, hogy hozzon létre egy 5 kocka széles tornyot valahol középtájon a **tolt** mezőbe tett egyesekből:

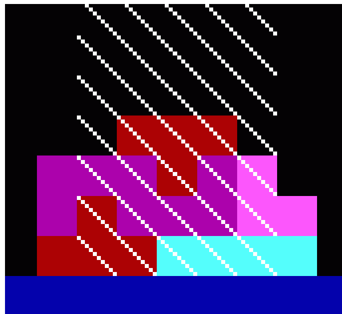
```
6 FOR x = 11 TO 15
akna(x, y).tolt = 1
NEXT
```

Tegyük láthatóvá az alakzatunkat. Nézzünk be **aknarajz**ba. Ez két egymásba ágyazott ciklussal végigmegy az aknán és mindegyik kockájánál meghívja **kockarajz**ot. Kérdés: ide, **kockarajz** hívása után tegyük be az alakzat megjelenítését, vagy inkább magába **kockarajz**ba? Jó lenne ez utóbbi, mert akkor a leeső elemen esés közben látszana, hogy melyik kockái vannak az alakzat területén és melyikek nem, de a **kockarajz** csak a koordinátákat és a színt kapja paraméterül, ezt esetleg ne változtassuk meg, ha nem muszáj. A programozó gondol a jövőre is: nem árt egy olyan rutin, ami minden megkötés nélkül tud olyan kockát rajzolni, amilyet az elemekhez használunk. Címlapot, a következőként várható elemet, hasonlókat megjeleníteni hasznos lesz.

Tehát az **aknarajz**ban, a **kockarajz** hívása után kell megrajzolnunk az alakzatot. Olyan minta kell, ami mögött látszanak a kockák. Egyelőre elég lesz egy áthúzás.

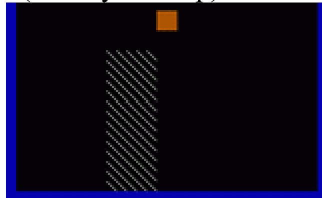
```
IF akna(x, y).tolt THEN LINE((x - 1) * kocka, (y -1) * kocka)-STEP(kocka - 1, kocka - 1), 15
```

Ha a kérdéses koordinátahely **tolt** mezője nem nulla, akkor vonalat húzunk a bal felső sarok képernyőre átszámított koordinátájától jobbra lefelé. Nézzük, mi lesz.



Kezdetnek nem rossz. Persze azt elfelejtettük közben, hogy ezt a rácsot nem akarjuk egészen az akna tetejéig fölvinni, mert ott nem lehet megtölteni kockákkal, de hát nem figyelhet az ember mindenre egyszerre. Egyelőre bőven elég, ha visszamegyünk az **inic**be és a 6–8. sor **FOR** ciklusát beletesszük egy **IF y > 5 THEN** utasításba.

Aknánk így ilyen lesz (kicsinyített kép):



A következő feladat a rácsot puszta rajzból működő valamivé tenni. Morfondírozzunk. Ha a ráccsal ellátott aknapozíciókat mind kockák fedik, akkor a játékos valami jutalmat kap. Ez a helyzet nyilván egy elem (az utolsó) lerakásakor állhat elő. Tehát a **ragadas** lesz a mi barátunk, ott kell beavatkoznunk. Legjobb, ha a legvégén, a **sorok** hívása után tesszük, mert akkor az elem elhelyezése után először eltűnnek a betöltött sorok, és csak aztán kaphatja meg a játékos a jutalmat (hacsak a betöltött sorok eltűnésével szabaddá nem válik a rács valamelyik része). A rács vizsgálata legyen külön rutin, nevezzük **racs**nak; ezt hívjuk itt meg.

Mi lesz a **racs** teendője? Végig kell néznie az aknát. Ha egy aknapozícióban rács van, akkor meg kell nézni, hogy van-e ott kocka. Ha csak egyetlen helyen nincs, akkor már nincs is miről beszélnünk, a rácsnak nincs minden mezője lefedve kockákkal. Ez ugyanaz a módszer, amit a **sorok** is alkalmaz: ha csak egyetlen oszlopban üres hely van, akkor a sor nincs betöltve. Lássuk:

```
DEFINT A-Z
SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt THEN IF akna(x, y).szin = 0 THEN EXIT SUB
NEXT: NEXT
```

Eddig megvagyunk, ellenőriztük a helyzetet. Ha csak egyetlen ráccsal fedett helyen nincs kocka, akkor kilépünk az egész rutinból, hagyjuk a dolgot a csodába. Ha mindkét ciklus lefutott, akkor a rács be van töltve. Egyelőre csak tegyünk ide egy **STOP**-ot és próbáljuk ki a programot. A dolog működik. Abban a pillanatban, hogy a rács utolsó helyét is betöltjük kockával, a program leáll ezen a helyen.

Találjuk ki, mit csináljunk, ha a rács be van töltve. Egy ötlet: rajzoljunk másik rácsot máshol. Ha az is betelik, csináljunk megint másik rácsot, és így tovább. Így a programunk olyan lesz, mint egy logikai játék, amiben különböző feladatokat kell teljesíteni, van első pálya, második pálya s így tovább.

Akkor viszont ki kell vennünk az **inic**ből a rács elkészítését, mert az csak a program indulásakor fut le, nekünk viszont később is kell rácsot rajzolnunk. Vegyük tehát ki, és írjunk helyette egy külön rutint a rács inicializálására; legyen a neve **racsinic**, és legyen egy paramétere: a rács sorszáma. Különböző rácsainkat majd ezzel különböztetjük meg egymástól.

Először is célszerű lesz eltüntetni minden rácsot, hogy amikor másodszor, harmadszor hajtjuk végre a rutint, ne maradjon ott a korábbi rács nyoma.

```
DEFINT A-Z
SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
akna(x, y).tolt = 0
NEXT: NEXT
```

Ha ez megvan, nézzük meg a paramétert és aszerint készítsünk rácsot. Egyelőre kezeljük le az **sz = 1** és **sz = 2** eseteket.

```
SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
```

Ebből egy függőleges oszlop lesz, kicsit betoltam középre, hogy szebb legyen. A második pedig legyen mondjuk két oszlop az akna két szélén.

```
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
```

```
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT
```

Persze ezt még meg is kell hívni, különben nincs rácsunk. Célszerű lesz megjegyezni, hogy melyik rács következik a sorban, mondjuk legyen erre egy változónk **rcs** néven, ezt deklaráljuk a főprogramban: **DIM SHARED rcs**, aztán az **inic**ben állítsuk be az értékét 1-re.

Mikor hívjuk meg a **racsinic**et? A program elején és amikor az előző rács betöltődött. Tehát kerüljön **inic** végére egy **racsinic rcs** utasítás, és egy ugyanilyen oda, ahol megállapítjuk az előző rács betelését, hol is van az… persze, a **racs** rutin végén a **STOP**-nál. Ezt vegyük ki és a helyére tegyük az utasítást. Ja igen, és írjuk oda elé, hogy **rcs = rcs + 1**, különben mindig ugyanazt a rácsot fogja létrehozni.

Van egy kis probléma. A játék nagyon jól működik odáig, hogy az első oszlopot megtöltjük kockákkal, és remekül megjelenik a két oszlop. Viszont ember legyen a talpán az a játékos, aki ezeket is meg tudja tölteni kockákkal, mert a képernyő közepén ott meredezik az oszlopunk és elállja az utat; a második-harmadik elem óhatatlanul bele fog rohanni és elérjük a képernyő tetejét. Ez így nem jó, a játékos így nincs motiválva arra, hogy megtöltse a rácsot, hiszen csak egy másik rácsot kap érte, az előző rács miatt feltornyozott elemek viszont ott maradnak. Jutalmazzuk azzal, hogy elvesszük ezeket. Bizonyos helyzetekben a játékos alig várja majd, hogy beteljen a rács és megszabaduljon a sok-sok kockától, ez ad majd izgalmat a játéknak; kicsit másmilyen izgalmat, mint a hagyományos, „rácstalan” tetrisekben. De *csak* azokat a kockákat vegyük el, amik a rács területén voltak. Ne legyen túl kényelmes a játékos helyzete.

Ez a **racsinic** rutin dolga lesz. Az elején úgyis nullázza az **akna** tömb **tolt** mezőjét, nem tart hát semeddig, hogy előbb megnézze, van-e törlendő kocka vagy sem. Az **akna(x, y).tolt = 0** sor helyére írjuk ezt:

```
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0
```

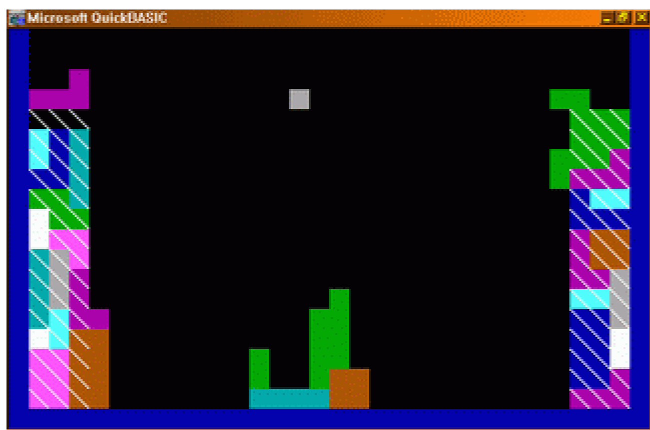
És nem fog ártani a rutin végén egy hívás az **aknarajz**ra.

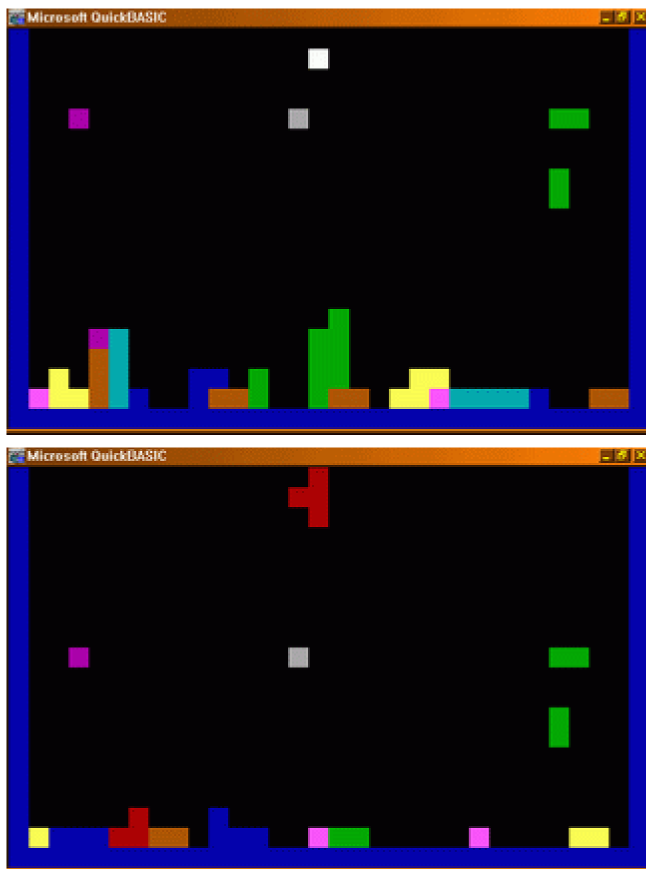
Sajátos tetrist hoztunk így össze. A játékos megtölti az első oszlopot, az eltűnik és hátrahagy némi szemetet a levegőben; ezután a játékos megtölti a két szélső oszlopot, azok is hagynak némi szemetet a levegőben; aztán nincs több rács, a játékos pedig nekiállhat eltüntetni a szemetet. Nem kis munka.

Csináltam néhány képet egy próbajátékról.


Microsoft QuickBASIC


Microsoft QuickBASIC


Microsoft QuickBASIC



Ezúttal elég kevés szemét maradt az első oszlop eltakarítása után: alul egy **Z** elem egyik fele és legfölül az építkezést befejező **H** csúcsa. Menet közben összeakadtam egy **V** elemmel, amit nem tudtam beilleszteni az oszlopba, úgyhogy a bal alsó sarokban tettem le, tudván, hogy később ott rács lesz és jól járok vele. Ez arra hasonlít, amikor egy kalandjátékban a többedszerre játszó játékos előnyben van a kezdővel szemben, tudja, mikor mire számíthat, tud előre tervezni. A kétoszlopos résznél már nem volt ilyen szerencsém, két **O** elem fele és egy **L** vége kimaradt bal oldalt alul, egy másik **L**-é az oszlop tetején; jobb oldalt két **Z** elemből maradtak ki részek; és egy csomó elemet le kellett tennem középen, mert nem volt nekik hely az oszlopokban. Ezt az építményt próbáltam eltüntetni a két utolsó kép készítése előtt, de amint a levegőben lebegő kockák helyváltoztatásából látszik, jó néhány sort be kellett töltenem, amíg az utolsó képen látható eredményt elértem.

Rácsaink tehát remekül működnek; a következő feladat az lenne, hogy a két oszlop betöltése után újabb rácsváltozatokat dolgozzunk ki, jó sokat, és mondjuk az utolsó rács betöltése után **rcs**-t 1-re visszaállítva elölről kezdődjenek a rácsok.

Ez már egy egész komplett játék lenne. Az Olvasónak módjában áll ezt a változatot kidolgozni, ha kedve van hozzá, de most másfelé invitálom.

```
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 11
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y).szin
IF akna(x, y).tolt THEN LINE ((x - 1) * kocka, (y - 1) * kocka)-STEP(kocka - 1, kocka - 1), 15
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
```

```
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 1
akna(aknax, y).szin = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
```

```
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) = 1 THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) = 1 THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN i = ii: x = x - 2: aknarajz
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt THEN IF akna(x, y).szin = 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB ragadas (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
```

```
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
NEXT
aknarajz
END SUB
```

Kétféle rács

A programozó elégedetten játszogat „berácsozott” tetrisével, és támad egy gondolata. Mi történne, ha nemcsak olyan mezők lennének, amiket be kell tölteni ahhoz, hogy új pályát kapjunk, hanem olyanok is, amiket viszont üresen kell hagyni hozzá? Vagyis nem szabad betölteni. Lennének kötelező mezők, amilyenek most vannak, és tilos mezők, ahol nem lehet kocka. Amíg bármely tilos mezőn kocka van, addig hiába töltjük be a kötelező mezőket, nem kapunk új rácsot; előbb el kell tüntetni a tilos mezőkön levő kockákat.

Ez igazán nem boszorkányság. A **tolt** mező egyessel jelzi a kötelező mezőket – hát vezessük be, hogy a tilos mezőket viszont kettessel jelölje. Először is hozzunk

létre egypár tilos mezőt, hogy lássuk, mivel dolgozunk. Legyen például néhány tilos mező az akna alulról a harmadik sorában, akkor nem okoz gondot, hogy alatta elhelyezzünk egypár máshova nem illő elemet, és könnyen be is tudjuk tölteni a tilos mezők némelyikét, hogy lássuk, csakugyan úgy hatnak-e, ahogy kell nekik.

Tegyük a racsinicbe ezt a részt:

```
13 FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
```

Ha elindítjuk a programot, egyből meglátjuk, hova kerültek a tilos mezők, mert ugyanúgy jelennek meg, mint a kötelezőek. Ne jelenjenek meg ugyanúgy. Menjünk **aknarajz**ba és változtassuk meg. Az **IF akna(x, y).tolt THEN** részből legyen egy **SELECT CASE akna(x, y).tolt** és egy **CASE 1**, ezután jöjjön a **LINE** utasítás. Másoljuk le ezt a sort és tegyük be még egyszer, írjuk át a **CASE** utáni 1-et 2-re, és biggyesszünk mögé egy **END SELECT**-et. Most van egy **SELECT CASE** elágazásunk, ami két szakasztott egyforma utasítás közül választ. Ne legyenek egyformák, különbözzön a vonalak színe. Legyen a tilos mező piros, a kötelező pedig zöld. A két szín kódja 12 és 10, ezek kerülnek a 15 helyére a **LINE** utasítások végén.

Eddig jó, tegyük a tilos mezőket csakugyan tilossá. Erről a **racs** rutinnak kell gondoskodnia, ez felelős a rács eltüntetéséért, ha minden elemét betöltöttük. 5. sora nézi meg, hogy a **tolt** mező nem nulla értéke mellett van-e kocka. Ezt ki kell javítanunk arra, hogy konkrétan az 1-es értéket figyelje. És kell mögé egy másik, szakasztott ugyanilyen sor, ami viszont a 2-es értéket figyeli, és ha ilyet talál, azt ellenőrzi, hogy a **szin** mező ugyanakkor nem nulla-e. A két sor együtt:

```
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
```



A próbajátékról készített képen látható, hogy a rács teljesen be van töltve, de nem tűnik el: egy **Z** elem felnyúló szára takarja az egyik piros vonalat. Ha azonban a legalsó sort betöltjük, az építmény eggyel lejjebb kerül, a piros rács már takaratlan, s elég betölteni a zöld oszlop legfelső sorát: az oszlop eltűnik és megjelenik helyette a két oldalsó oszlop.

```
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 11
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y).szin
SELECT CASE akna(x, y).tolt
CASE 1: LINE ((x - 1) * kocka, (y - 1) * kocka)-STEP(kocka - 1, kocka - 1), 10
CASE 2: LINE ((x - 1) * kocka, (y - 1) * kocka)-STEP(kocka - 1, kocka - 1), 12
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
```

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 1
akna(aknax, y).szin = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
```

```
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) = 1 THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) = 1 THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN i = ii: x = x - 2: aknarajz
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB ragadas (e, i, ex, ey)
```

```
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
NEXT
aknarajz
END SUB
```

Extrák

Egy igazi jó tetrisben *extrák* is vannak. Mik az extrák? Speciális elemek, amik valamilyen segítséget jelentenek a játékosnak, de az is előfordulhat, hogy hátráltatják a dolgát. Képzeljen el az Olvasó egy elemet, ami ugyanúgy érkezik, mint bármely más tetriselem, ugyanúgy le lehet tenni, de ahelyett, hogy ott elhelyezkedne, valami egészen mást csinál. Például egy kocka (□) méretű extra eltávolítja azt a kockát az aknából, amire rádobtuk. Nevezhetjük bombának is. Nagyon hasznos,

ha utat akarunk vágni egy betemetett lyukhoz, ha viszont nincsen ilyesmink, akkor semmit sem ér. Inkább bosszúságot jelent, hogy nem akkor jött, amikor szükség volt rá.

Morfondírozzunk. Hogyan különböztessük meg az extra kockát a rendes kockától? Amikor a **kockarajz** megkapja a kirajzolandó kocka adatait, tudnia kell, hogy extra kockát kell rajzolnia, de neki a koordinátákon kívül csak a színt adjuk meg. Hát legyen a szín az extra kocka jele. 256 színünk van, eddig felhasználtunk tizenkétfélét (a tizenegy elem színeit és a feketét), bőven van miből válogatni. Jelentsék bizonyos színek azt, hogy az a kocka extra, mondjuk a 255-ös szín jelentse azt, hogy az imént megbeszélt bombáról van szó. Ez nem azt jelenti, hogy ez a kocka 255-ös színben fog megjelenni, először is azért nem, mert az fekete (ha át nem definiáljuk), másrészt pedig mert az extra kockának nem célszerű pusztán a színében különböznie a nem extra kockától. Ha a véletlen úgy hozza, hogy a játékos előbb kap extra kockát, mint ugyanolyan küllemű rendeset, akkor azt fogja hinni, hogy annak az elemnek az a rendes színe, és meg lesz lepve, amikor az elem földet érve valami szokatlant csinál. Ezért inkább úgy csináljuk, hogy a 255-ös szín csak egy jelzés legyen a **kockarajz** számára, ebből tudja, hogy nem normál kockát, hanem speciális alakzatot kell rajzolnia, és azt rajzol.

Mekkora legyen az elemünk? Ha csak egyetlen kockából áll, akkor nem kell azzal foglalkoznunk, hogy a több kocka közül mindegyiknek meglegyen-e a speciális hatása vagy némelyiknek ne, akkor az elemünk egyetlen extra hatású kockából áll és kész, egyszerűbb a helyzet.

Hogyan tároljuk? Az **ujelem**nek ki kell sorsolnia, hogy most extra következik, az **elemrajz**nak ki kell rajzolnia, a **mozgas**nak mozgatnia kell a **mehet** ellenőrzésével, egyszóval mindaddig, amíg hatását ki nem fejti, ugyanúgy kell viselkednie, mint egy köznapi tetriselemnek.

Az egyik megoldás az lenne, hogy vesszük a **K** (□) elemet, és valahol beállítunk egy jelzést, hogy vigyázat, most extra az illető. Ez azonban azt okozná, hogy a **K** elemek számbelileg hátrányba kerülnének az összes többivel szemben: ha **T** vagy **L** elemünk van, akkor olyan elemünk van, ha viszont **K,** akkor vagy **K,** vagy extra, mindegy, hogy mekkora az esélye ez utóbbinak, annyival kisebb lesz az esélye az előbbinek. Ez nem az igazi. Inkább vezessünk be egy külön elemet, aminek ugyanaz a ceruzakódja, mint a **K**-nak, de a színe 255-ös. Tegyük be egy **DATA**-sorba, növeljük meg eggyel **elemek**et és morfondírozzunk tovább.

Azáltal, hogy ezt megcsináltuk, lett két **K** elemünk, egy fehér és egy 255-ös színű, azaz fekete. Ha elindítjuk a programot, előbb-utóbb be is fog következni, hogy nem jelenik meg semmilyen leeső elem, de ha rávisszük a rácsra, akkor eltűnik egy-egy vonaldarab. Tehát a különbség végig megvan. Értesítsük a **kockarajz**ot, hogy ha 255-ös színt kap, akkor teljesen másképpen kell viselkednie. Most így szól a szövege:

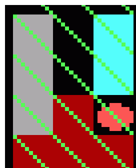
```
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), szin, BF
```

Az értékadásokat ne bántsuk, azok jók úgy, ahogy vannak. A **LINE** utasítást viszont tegyük feltételes módba:

```
IF szin <> 255 THEN
LINE (px, py)-STEP(9, 9), szin, BF
```

és írjunk valamit **ELSE** esetére. Mondjuk legyen az extra kocka arról felismerhető, hogy gömbölyű, nem szögletes. Élénk, figyelemfelkeltő színű, mondjuk világospiros. Egy piros golyó.

```
ELSE
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
END IF
```



Egész jól néz ki. Persze egyelőre semmi extrát nem csinál, csak helyet foglal az építményünkben, mint egy közönséges kocka, de az extraság most jön.

Maradjunk annál, hogy a golyó feladata kirobbantani azt a kockát, amire rádobjuk. Ez logikusan akkor következik be, amikor a golyó földet ér, vagyis a **ragadas**ba kell belenyúlnunk. Legjobb, ha külön rutint írunk az extra kezelésére, és ezt rögtön a **ragadas** elején meghívjuk:

```
IF szin(e) = 255 THEN extra e, i, ex, ey: EXIT SUB
```

Azért kell az **EXIT SUB**, mert az extra nem kövül meg az aknában, az egész ragadási folyamatot ki kell hagyni.

Csináljuk meg az **extra** rutint. Feladata nagyon egyszerű: az elem helye alatti kockát eltünteti, ha az nem az akna feneke. (Abba talán mégse üssön lyukat.)

```
DEFINT A-Z
SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN akna(ex, ey + 1).szin = 0: aknarajz
END SUB
```

Az **aknay - 1**. sor az akna feneke fölötti sor; ha az elemünk **ey** koordinátája kisebb ennél, akkor eggyel alatta nem az akna feneke van és robbanthatunk. Tehát nullázzuk az alatta levő elemet és újra kirajzoljuk az aknát.

Játékszerünk tökéletesen működik, s érdekes új színnel gazdagítja a játékot. Segít eltüntetni az első rács betöltése után az oszlop körül maradó, a mozgást akadályozó szemetet. Ha valamiért kénytelenek voltunk letakarni a piros rács egy részét, eltünteti az oda került kockákat és felszabadítja a rácsot.

Azazhogy ez utóbbival van egy kis gond. Ha az a szituáció áll elő, hogy a játékos teljesen betölti a zöld rácsot, és ugyanakkor a piros rácsból is le van fedve egy kocka, aztán ezt golyóval kirobbantjuk, akkor ott áll a teljesen betöltött zöld rács plusz a teljesen szabad piros rács, és a program nem reagál. A zöld rács csak akkor tűnik el és viszi magával a kockákat, amikor *a következő* elemet lerakjuk. Nem szép.

Csakugyan, hiszen a rács betöltöttségét a **racs** rutin ellenőrzi, ezt pedig a **ragadas** legvégén hívtuk meg; ennek a rutinnak azonban a teljes szövegét ki-

hagytuk, ha extra jött. Írjuk tehát oda a **racs** hívását az **extra** egyetlen sorának végére.

```
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13
DATA k, k, k, k, 255

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
kockarajz x, y, akna(x, y).szin
SELECT CASE akna(x, y).tolt
CASE 1: LINE ((x - 1) * kocka, (y - 1) * kocka)-STEP(kocka - 1, kocka - 1), 10
CASE 2: LINE ((x - 1) * kocka, (y - 1) * kocka)-STEP(kocka - 1, kocka - 1), 12
END SELECT
```

```
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN akna(ex, ey + 1).szin = 0: aknarajz: racs
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 1
akna(aknax, y).szin = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
IF szin <> 255 THEN
LINE (px, py)-STEP(9, 9), szin, BF
ELSE
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
END IF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
```

```
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K": IF mehet(e, i, x - 1, y) = 1 THEN x = x - 1: aknarajz
CASE CHR$(0) + "M": IF mehet(e, i, x + 1, y) = 1 THEN x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN i = ii: x = x - 2: aknarajz
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
```

```
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB ragadas (e, i, ex, ey)
IF szin(e) = 255 THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
```

```
NEXT
aknarajz
END SUB
```

Minták

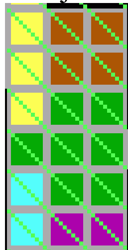
Hat fejezettel ezelőtt azért tértünk át karakteres képernyőről grafikusra, mert akkor több színt használhatunk és mintásak lehetnek a kockáink. Azóta egy ujjunkat se mozdítottuk ez ügyben, ami egyáltalán nem helyes dolog. Csinosítsuk ki a programot!

Ha például a **kockarajz**ban a **LINE (px, py)-STEP(9, 9), szin, BF** utasítás után beteszünk egy **LINE (px, py)-STEP(9, 9), 7, B**-t is, akkor a kockáinknak világosszürke kerete lesz. Sajnos a fekete kockáknak, vagyis az akna üres részének is. Írjuk a sor elejére gyorsan, hogy **IF szin <> 0 THEN**.

Nem tökéletes: a körvonalak vibrálnak. Persze, a **mozgas** folyamatosan újrarajzoltatja az elemet, és a keretet a már berajzolt részre rajzoltuk rá. Vagyis megjelenik (mondjuk **T** elemnél) a piros négyzet, rákerül a szürke keret, azt felülírja a piros négyzet, megint a keret… Javítsuk ezt ki. Az első **LINE** utasításban álljon **px + 1, py + 1** és **STEP(7, 7)**.

Nagyszerű, nem vibrál, éppen csak otthagyja maga után a szürke kereteket. Miért is? Mert amikor törlünk, akkor a belső részt feketével festjük ki, de a szürke keretet minden körülmények között szürkével rajzoljuk. Az előbb még letörölte a keretet, amikor feketével töltötte ki a teljes négyzetet, de ezt most már nem teszi.

Írjuk a 6. sor végére: **ELSE LINE (px, py)-STEP(9, 9), 0, B**. Így majd letörli.



Most már minden jó, egy gond van csupán. A zöld vonalkázás nem nagyon érvényesül a kockák új színezésénél. Eddig se volt egy szépség, de most már kifejezetten snassz. Tegyük szebbé.

Továbbra is úgy kellene, hogy a rács mintája helyenként átlátszó legyen, lehessen alatta látni a kockát. Mi lenne, ha sakktáblamintát kapna? Minden második pontot bezöldítenénk (illetve a tilos mezőknél bepirosítanánk), a többi pontban pedig látszana a kocka színe. Próbáljuk ki.

Az **aknarajz**ban van a rács kirajzolása. Kezdjük a **CASE 1**-gyel, töröljük ki, ami mögötte van, és írjuk ide a zöld rács új változatát.

Morfondírozzunk. Sakktáblamintát úgy csinálunk, hogy két egymásba ágyazott ciklussal végigmegyünk a területen, és megvizsgáljuk, melyik pontot milyen színűre kell festeni. Az egyik sorban minden második pontot ki kell gyújtani, mondjuk azokat, amelyeknek az oszlopkoordinátája páros, a másikban viszont éppen ellenkezőleg, a páratlanokat kell kigyújtani. Mondjuk csináljuk úgy, hogy ha mindkét koordináta páros, akkor kigyújtjuk a pontot, különben nem.

Mettől meddig is fog menni a ciklusunk? Hozzuk csak ide a **kockarajz** első sorát, itt is ugyanazokra a koordinátákra van szükség, tegyük be az **aknarajz** leg-

elejére. Tehát a ciklusunk **px**-től **px + 9**-ig fut az egyik irányban és **py**-tól **py + 9**-ig a másikban.

Hogyan nézzük meg, hogy mindkét koordináta páros-e? Ha két szám páros, akkor az összegük is páros. Ha két szám páratlan, az összegük akkor is páros. Ha két szám közül az egyik páratlan, a másik páros, akkor az összegük páratlan. Vagyis ha összeadjuk a két számot és az összegük páratlan, akkor különböző paritásúak, máskülönben egyezőek. Oké, gyűjtsuk ki a pontokat ebben az esetben.

```
FOR rx = px TO px + 9
FOR ry = py TO py + 9
IF (rx + ry) / 2 = (rx + ry) \ 2 THEN PSET (rx, ry), 10
NEXT: NEXT
```

Ezt olyan jól csináltuk, hogy az égvilágon semmi nem jelenik meg, fekete marad a rács. Debugoljunk. Mennyi **px** és **py**?… persze, ezek értékét **x**-ből és **y**-ból számítjuk ki, azoknak viszont semmiféle értéke nincs a rutin legelején. Később kapnak értéket, tegyük csak át szépen az értékadásokat a rutin elejéről a **SELECT CASE** elé.

Most már jó, azaz nem jó, nagyon vibrál. Szép zöld rácsot kaptunk, de vibrál. Miért is? Mert előbb kirajzolja a fekete kockát, aztán a rácsot, megint a fekete kockát, megint a rácsot… Csináljunk ezzel valamit. Semmiféle rajzolási technika nem tudja kiküszöbölni a vibrálást, ha folyton letöröljük és újrarajzoljuk ugyanazt. Ne tegyük. Ha a kocka fekete, akkor a **kockarajz** egyszerűen ne rajzolja ki. Tegyük az 5. sorának elejére, hogy **IF szin THEN**.

Most már totál rossz. Még mindig vibrál, és ahogy mozog az elem, ott maradnak a kockák belső, színes részei. Azokat nem törli le a **kockarajz**. Vegyük ki ezt az **IF szin THEN** részt, és tegyük inkább az **aknarajz**ba, a **kockarajz**ot hívó sor elejére, hogy **IF akna(x, y).szin THEN**.

A vibrálást megszüntettük, viszont most már semmi se törlődik le az elemből mozdítás után. Ez azért mégiscsak tisztább helyzet, innen könnyebb elindulni.

A vibrálás nyilvánvalóan azért szűnt meg, mert ha egy helyen az **akna** tömb **szin** mezője 0-t tartalmaz, akkor **kockarajz** most már egyáltalán nem hajtódik végre és nem törli le a korábban odarajzolt dolgokat, vagyis a rácsot. De az eső elemet sem, mivelhogy azt mostanáig szintén a **kockarajz** törölte. Általában: a **kockarajz** törölt minden olyasmit, ami üres mezőben van és nem kocka.

Ha gondoskodunk az eső elem törléséről mozdítás után, akkor megoldottuk a problémát. Legjobb lesz külön csinálni egy rutint erre a célra. Legyen a neve **elemtorles**, és másoljuk bele az **elemrajz** szövegét. Ha ez megvan, a **kockarajz** hívását vegyük ki, mert az úgyse rajzolna semmit (hiszen az elemünk mindig üres mezőkön sétál), helyette tegyünk egyszerűen egy **LINE (px, py)-STEP(9, 9), 0, BF** utasítást. S hogy ez működjön is, a **kockarajz** elejéről másoljuk a **LINE** elé a **px**-et és **py**-t értékkel ellátó sort is.

Hol hívjuk meg **elemtorles**t? Ez **mozgas** valamelyik pontján lesz célszerű, ott, ahol elmozdítjuk az elemet. Sajnos nincs egyetlen ilyen pont a rutinban, mert

a mozdításhoz szükséges döntések miatt szétágazik a végrehajtás; kénytelenek vagyunk a **mehet** függvényt kiértékelő **IF**-ek **THEN**-jei mögé írni az **elemtorles e, i, x, y** utasításokat. Nyolc példányra lesz szükség. Vigyázat: a **CASE 1** kezdetű sorba is kell egy, ahol semmilyen **IF** nem áll.

Kicsit át is tördeltem a rutin szövegét, úgyhogy most így néz ki:

```
SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1: aknarajz
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2: aknarajz
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB
```

Elemünk most már szépen letörlődik mindig, és a rács sem vibrál. A lerakott elem látszik a rács mögött.

A piros rácsot is ugyanígy kellene megcsinálni. Menjünk csak vissza **aknarajz**ba.

A zöld rács kirajzolását két **FOR** ciklus végezte. Miért másoljuk ezt a négysoros programrészt a **CASE 2** mögé, ha lehet máshogyan is? Elvégre a kétféle rács között csak a színben van különbség. Ha **CASE 1** helyett azt írjuk: **CASE 1, 2**, ak-

kor mindkét rács kirajzolását ugyanaz a programrész végzi majd, és **akna(x, y).tolt**ben megtaláljuk, hogy éppen melyikről van szó. Ha értéke 1, akkor 10-es színnel rajzolunk; ha 2, akkor 12-essel. A feladat csak annyi, hogy az 1 és 2 számokat a 10 és 12 számokká alakítsuk át. 1 és 2 között 1 a különbség, 10 és 12 között viszont kettő, hát akkor szorozzuk be a számunkat kettővel; így lesz 2 és 4. Ha hozzáadunk 8-at, akkor készen is vagyunk.

Már egész jó, a piros rácsunk is rácsszerű, csak egy probléma van: a bomba nem működik. Ledobás után ott marad a bomba is, a kirobbantott kocka is. Persze, mert mindkét helyen 0 van a **szin** mezőben, és ilyenkor a **kockarajz** nem csinál semmit. Töröljük ezeket a mezőket az **extra** rutinból. De ne tegyünk bele két **LINE** utasítást, ha nem muszáj, hanem vegyük ki **elemtorles**ből az effektív törlési műveletet, tegyük be egy rutinba, ami így nézhet ki:

```
DEFINT A-Z
SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB
```

Ezt hívjuk meg **elemtorles**ből és **extrá**ból is. Sokkal kényelmesebb lesz.

Még mindig nem vagyunk készen. A rács továbbra is vibrál, csak nem az üres részen, hanem ott, ahol már van mögötte kocka. Csakugyan, hiszen itt minden lépésben kirajzoljuk a kockát, aztán újrarajzoljuk a rácsot. Kellemetlen. Ráadásul amikor a rácsot betöltöttük, az egyáltalán nem tűnik el, ott marad, és az arra vitt elemek törölgetik le. A vibrálás várhat, ezt viszont azonnal meg kell csinálni.

A **racsinic** rutin végzi a betöltött rács eltüntetését, tartalmával együtt. Ide tegyük be **kockatorles** meghívását, az 5. sor végén.

Most már jó, a rács eltűnik.

A programozó gyanút fog és nekiáll betölteni egy sort. Hát persze hogy nem tűnik el!

Menjünk hát a **zuditas** rutinba, és a **szin** mezőbe 0-t töltő 8. sor után írjunk be egy **kockatorles x, 1**-et.

Nem elég. Hát persze, ez csak a legfelső sort törli. Akkor tegyünk be 6. sornak egy **kockatorles x, yy**-t.

Ez végre már megteszi hatását, a kép rendesen újrarajzolódik.

Pihenjünk egy kicsit. Kellemetlen volt ez a hosszú hibaüldözés, mi nem akartunk semmi mást, csak szebb grafikát. Ez már csak így szokott lenni programozás közben.

```
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13
DATA k, k, k, k, 255

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE akna(x, y).tolt
CASE 1, 2: FOR rx = px TO px + 9
FOR ry = py TO py + 9
IF (rx + ry) / 2 = (rx + ry) \ 2 THEN PSET (rx, ry), akna(x, y).tolt * 2 + 8
NEXT: NEXT
```

```
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockatorles x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 1
akna(aknax, y).szin = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT
```

```
RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
IF szin <> 255 THEN
LINE (px + 1, py + 1)-STEP(7, 7), szin, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 7, B ELSE LINE (px, py)-STEP(9, 9), 0, B
ELSE
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
END IF
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1: aknarajz
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1: aknarajz
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii: aknarajz
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1: aknarajz
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1: aknarajz
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2: aknarajz
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2: aknarajz
```

```
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1: aknarajz
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB ragadas (e, i, ex, ey)
IF szin(e) = 255 THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
```

```
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
kockatorles x, 1
NEXT
aknarajz
END SUB
```

Folytatás

Hozzuk rendbe programunk háza táját! A rács betöltött része vibrál, az elem időnként egészen hosszú időre eltűnik mozgatás közben, és (főleg amikor már két rácsoszlopunk van) „lendülettel" mozog, vagyis már elengedtük a gombot és még mindig halad. Ez így nem lesz jó.

Miért is csinálja mindezt? A vibrálást már megbeszéltük, a másik kettőt pedig a grafika lassúsága miatt. Túl sokáig tart kirajzolni a képet, emiatt a sokáig lenyomva tartott nyílbillentyű azt okozza, hogy mire elkészül a kép kirajzolásával, addigra a billentyűzetpufferben megint megtalálja a billentyű kódját, végrehajtja, mire elkészül vele, megint ott találja, aztán amikor végül fölengedjük a billentyűt, már egy csomószor bekerült a kód a pufferbe, egy csomó lépésbe telik, amíg a program kiszedegeti és végrehajtja.

Gyorsítsuk föl a grafikát. Az **aknarajz** rutin a fél képernyőt telerajzolja kockákkal, néhol kétszer is (a rács betöltött részén), minek ezt minden lépésben meg-

csinálni? Eredetileg azért hívogattuk folyton az **aknarajz**ot, hogy mozdítás után letörölje az elemünk korábbi képét, de ezt többé már nem teszi, az **elemtorles** csinálja helyette. Amíg az elem csak mozog a képernyőn, az aknát egyáltalán nem kell újrarajzolni. A **mozgas**ból vegyük ki az **aknarajz** összes hívását. Nyolc helyen szerepel.

Grafikánk csodálatosképpen megjavult. Az elem többé nem tűnik el mozgatás közben, mert a gépnek most már sokkal kevesebb a rajzolnivalója és azt van ideje újrarajzolni. Nem szalad el az elem, és mintha a rács betöltött része se vibrálna. Ez utóbbit legjobban úgy lehet ellenőrizni, hogy betöltünk valamennyit a rácsból, és egy elemet folyamatosan forgatni kezdünk. Ez a művelet ugyanis bármeddig ismételhető (ha az elemnek van helye forogni), a mozgatások viszont csak egy ideig. Tényleg nem vibrál. Nem is teheti, hiszen amíg az elem el nem helyezkedett az aknában, egyáltalán nem rajzoljuk újra annak tartalmát.

Ja, hogy az elemek menet közben letörlik a rácsot… Hát azt senki se mondta, hogy minden tökéletes lesz.

Javítsuk ezt ki. Miért törli az elem a rácsot? Mert amikor valahonnan eltávozik, akkor az **elemtorles** csak fekete kockákat rak a helyére, és az **aknarajz** végrehajtása elmaradván senki se rakja vissza a rácsot. Meg kell beszélnünk **elemtorles**sel, hogy rakja vissza ő.

Ha két helyen rajzoljuk ugyanazt, akkor érdemes rutint csinálni rá. Tegyük a rács kirajzolásával is ezt. Vegyük ki **aknarajz**ból a **CASE 1, 2**-höz tartozó részt, és tegyük be egy **racsrajz** nevű új rutinba, aminek **x** és **y** legyenek a paraméterei. Hozzuk át ide a **px**-nek és **py**-nak értéket adó sort is. **Aknarajz**ban a **CASE 1, 2** után pedig írjuk ezt: **racsrajz x, y**.

Következik **elemtorles**. Ebben van egy **kockatorles** utasítás, ez után érdemes meghívni **racsrajz**ot, hogy a rács a törölt kockára rajzolódjon; de csak akkor, ha van rács azon a mezőn, mert maga a **racsrajz** nem ellenőriz semmit, mindenképpen rajzol.

CASE "k": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y

A program ismét tökéletesen dolgozik.

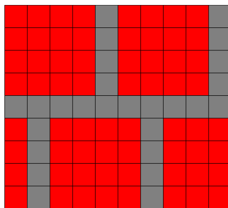
Mindaddig, amíg egy bombát véletlenül nem az akna fenekén dobunk le, mert akkor az bizony ott marad. Igen, mert az **extra** rutin csak akkor csinál bármit is, ha a bombát kockára dobtuk. Írjuk bele, hogy **ELSE aknarajz**.

Nem jó. Persze, mert ez a rutin nem töröl többé, csak rajzol. Legyen **ELSE kockatorles ex, ey**.

Most jó!

Visszatérhetünk korábbi tevékenységünkhöz, a grafika mintássá tételéhez, amit a rajzolási hibák megakasztottak. Először is szüntessük meg azt az állapotot, hogy az akna falai teljesen ugyanúgy néznek ki, mint a kockák, éppenséggel az **L** elemmel még színben is megegyeznek. Ronda, snassz, vacak. Legyenek olyanok, hogy első ránézésre lehessen tudni, melyik a kocka és melyik a fal.

Mondjuk legyen téglafal. Azt nem nehéz rajzolni. Egy kocka ugye 10·10-es pontrácsból áll, tervezzünk ilyenben téglamintát. Csak egy darab kockás papír kell.



Téglamintánk úgy készül, hogy a tíz képpontnyi magasságon elhelyezünk két sor téglát, mindkettő öt-öt képpont magas lesz; ebből egy a köztük levő vízszintes vonal, a többi négy a tégla. Ugyanígy csináljuk vízszintesen is, négy pont széles a tégla és egy pont a függőleges vonal, de egymáshoz képest elcsúsztatjuk a két téglasort.

Eddig jó, csináljuk is meg. Nyilván a **kockarajz** fogja kirajzolni a téglákat is, de honnan tudja, hogy mikor rajzoljon olyat? Hát a színkódról. Ahogy a 255-ös szín azt jelentette, hogy piros golyót kell rajzolni, egy másik szám jelentse azt, hogy téglákat kell. Legyen mondjuk a 100-as színkód a téglák jele. Az akna falát az **inic** alakítja ki, mindmáig az egykor „van kocka” jelentésű 1-esek kerülnek bele – hát javítsuk ki őket 100-ra. A rutin 4., 5. és 8. sorának végére írjuk ezt a számot.

Az akna fala világoskék lett, a 100-as szín ugyanis jelenleg ilyen. Menjünk a **kockarajz**ba és gondoskodjunk róla, hogy ezt a számot ezentúl téglasorként ábrázolja.

A rutin jelenleg megvizsgálja, hogy a színkódunk 255-e, és eszerint hajt végre kétféle rajzolási műveletet (kocka – golyó). Ezt jobb lesz átalakítani **SELECT CASE**-zé.

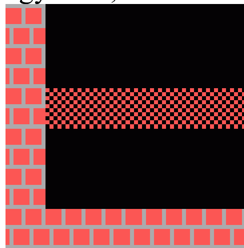
```
4 SELECT CASE szin
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 7, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
```

Az **ELSE** ág lesz a kockarajzolás, ha nem olyan színt kaptunk, ami valamilyen speciális dologra utasít, akkor kockaszínt kaptunk és kockát rajzolunk. A **CASE 100** részt célszerűen a **CASE 255** elé tegyük, a számok sorrendjében.

A tégla leírása a négyzetrácsból könnyen kiolvasható. Rajzolnunk kell pirossal három darab 4·4-es négyzetet, egy 4·3-as téglalapot és egy 4 pont hosszú függőleges vonalat; szürkével pedig két vízszintes vonalat és négy függőlegest. Kezdjük a pirosakkal. Számolgassuk ki a koordinátákat és írjuk meg az utasításokat.

```
CASE 100
LINE (px, py)-STEP(3, 3), 12, BF
LINE (px + 5, py)-STEP(3, 3), 12, BF
LINE (px + 2, py + 5)-STEP(3, 3), 12, BF
LINE (px + 7, py + 5)-STEP(2, 3), 12, BF
LINE (px, py + 5)-STEP(0, 3), 12
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
```

Az első utasítás a bal felső piros négyzetet rajzolja meg, a második a jobb felsőt, a harmadik az alsót, a negyedik a jobb alsó téglalapot, az ötödik pedig bal oldalt alul a piros vonalat. Idáig pirossal dolgoztunk. A szürkével rajzoló utasítások közül az első kettő a felső és az alsó vízszintes vonalat húzza meg, a következő kettő a felső sor függőleges vonalait, az utolsó kettő az alsó sorbelieket. Igazán egyszerű, csak leolvasgattuk a koordinátákat és kész volt.



Nagyon szép. Talán egy kicsit még szebb lenne, ha a téglák és a malter között nagyobb lenne a kontraszt, így kissé egybefolynak. És ha a téglák nem pirosak lennének, hanem téglaszínűek?

Megpróbálhatunk tetszetősebb színeket kikeresni a grafikus kártya 256 alapértelmezés szerinti színéből, de sokkal jobb, ha biztosra megyünk: csinálunk magunknak saját palettát.

Hogyan készít a programozó palettát a játékprogramjához? Egyszerűen. Fogja a színkódokat 0-tól 255-ig, és tetszése szerint fölosztja tartományokra, innentől idáig ezeket a színeket használjuk, onnantól odáig amazokat, és így tovább.

Ha palettát készítünk, akkor alkalmasint jobb lesz az elemek színeit is átrendezni. Hülye dolog, hogy az 1-től 15-ig terjedő színek közül 13-at használunk, ráadásul teljesen összevissza. Már csak azért is, mert (a programozóban hirtelen fölmerül egy gondolat) később meg lehetne csinálni azt is, hogy a kockák egy kicsit térbeli ábrázolást kapjanak, valahogy úgy, mint a Windowsban a gombok, az egyik szélük világosabb, a másik sötétebb, de ehhez persze kell ugyanannak a színnek három árnyalata. Akkor foglaljunk le a kockák számára háromszor annyi színt, amennyi van. De ezt ne 1-től kezdjük, mert akkor most nyomban át kell szerveznünk a kockák színezését, hanem 16-tól, onnantól kezdve teljesen érintetlen a paletta. 12 kockaszínünk van, 3·12 az 36, akkor 16-tól 51-ig lefoglaltuk a palettát a kockák színei számára. Majd később kimorfondírozzuk, hogy ez részletesen hogyan történjen, most menjünk tovább.

A kockákon kívül e pillanatban a rácsok, a golyó és a téglafal vannak kirajzolnivalónak. Egyáltalán nem szükségszerű, hogy ezek annyi színt használjanak,

amennyit most használnak. Akárhogyan áttervezhetjük őket. Mondjuk a golyó is lehetne kicsit térbeli, az három szín, a rácsoknak most kell kettő, növeljük meg négyre a biztonság kedvéért, a téglafalnak egyelőre elég lesz a mostani kettő. Ez eddig kilenc szín. De álljunk csak meg egy szóra.

A golyó most ugyanolyan színű, mint a téglafalban a téglák és mint a tilos rács piros pontjai. Miért ne maradhatna ez így? Miért használjunk három színkódot ugyanarra a színre csak azért, mert más-más célokat szolgál?

Másrészt viszont miért *ne* használjunk három színkódot? Elvégre telik a palettából, és így meglesz az az előnyünk, hogy ha meg akarjuk változtatni a golyó színét, akkor csak igazítunk a paletta egyik színén, nem változik vele semmi másnak a színe, csak a golyóé. De mi van, ha egyszer csak nem telik már többé a palettából, mert annyi mindenfélét rajzoltunk már, mindent más-más színkódokkal?

Mindkét megoldásnak megvannak az előnyei és a hátrányai. Érdemes lesz többet is morfondírozni erről a palettáról.

```
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED szin(1 TO elemek)
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
```

```
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13
DATA k, k, k, k, 255

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
```

```
racs
ELSE kockatorles ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 100
akna(aknax, y).szin = 100
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 100
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 100
LINE (px, py)-STEP(3, 3), 12, BF
LINE (px + 5, py)-STEP(3, 3), 12, BF
LINE (px + 2, py + 5)-STEP(3, 3), 12, BF
LINE (px + 7, py + 5)-STEP(2, 3), 12, BF
LINE (px, py + 5)-STEP(0, 3), 12
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 7, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
```

```
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT
```

```
SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB racsrajz (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
FOR rx = px TO px + 9
FOR ry = py TO py + 9
IF (rx + ry) / 2 = (rx + ry) \ 2 THEN PSET (rx, ry), akna(x, y).tolt * 2 + 8
NEXT: NEXT
END SUB

SUB ragadas (e, i, ex, ey)
IF szin(e) = 255 THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

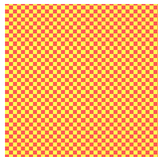
SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
```

```
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
kockatorles x, 1
NEXT
aknarajz
END SUB
```

Paletta

A morfondírozást legjobban az segíti, ha az ember közben a programjával játszogat. Általában amikor a programozó „programozás helyett” lelkesen játszik az általa éppen írt programmal, akkor valójában nagyon is elmélyülten dolgozik, a munkának azt a bizonyos 99%-át végzi. Most éppen az jut eszébe, hogy a zöld rácsra dobált elemek olyasféle benyomást keltenek, mintha az elem színe és a rács színe keveredne egymással. Ez nem csoda: ha egy területet sakktáblaszerűen betöltünk piros és sárga pontokkal, az kicsit távolabbról narancssárgának fog látszani:



A rácsnál is ezt tettük, sakktáblaszerűen betöltöttük zöld és másmilyen színű pontokkal, ez adja a színkeverés hatását. Mi lenne, ha *tényleg* kevernénk a színeket, nemcsak úgy tennénk, mintha?

A következőképpen képzeltem el a dolgot. Van egy zöld rácsunk, ami most már nem rácsszerű, hanem teljesen egyszínű zöld, nincs benne egyetlen fekete pont sem. Persze nem olyan élénk világoszöld, mint most, hanem jóval sötétebb. A rács mögé érkezik egy kocka, mondjuk piros színű. Ekkor kiderül, hogy a zöld felület áttetsző, mert mögötte látszik a piros kocka, de mivel egy zöld színszűrőn át látjuk, megsárgul-megbarnul a kocka. Két színt összekeverni nagyon könnyű, ha saját palettát használunk: tudjuk, milyen színösszetevőkből áll az egyik szín, milyenekből a másik, ezeket összekeverjük és hozzárendeljük az egyik színkódhoz. Van ugye tizenkét kockánk, mindegyikhez kigondoltunk három színt, ez harminchat színkód. Most vegyünk még harminchat színkódot, másoljuk oda ugyanazokat a színeket, de mindegyiket valamicskével több zölddel. Vegyünk még egyszer harminchat színkódot, oda is másoljuk az eredeti színeket, de valamicskével több pirossal.

Mondjuk legyen így: 16-tól 51-ig a rendes kockaszínek; 52-től 87-ig a zöldesített színek; 88-tól 123-ig pedig a pirosított színek.

Ezek után csak annyi a dolgunk, hogy amikor kirajzolunk egy kockát, megnézzük, hogy van-e ott rács, és ha igen, akkor a rendes színkódok helyett a zöld vagy a piros tartományból vesszük a színkódjainkat. Ez is nagyon könnyű: vesszük **akna(x, y).tolt** értékét, az lehet 0, 1 vagy 2. Beszorozzuk 36-tal, akkor lesz 0, 36 vagy 72. Hozzáadunk 16-ot, lesz 16, 52 vagy 88, ez a szükséges számtartomány kezdőpontja, eltesszük egy változóban és csak hozzá kell adni a kiválasztott szín kódjához, amit majd 0-tól 35-ig terjedően adunk meg.

Most lássuk azt a bizonyos harminchat színt. Van tizenkét elemünk, mindegyiknek egy normál, egy világosabb és egy sötétebb árnyalata. Legjobb lesz egymás mellé csoportosítani őket, sötét-közepes-világos. Tehát a harminchat színünk így épül fel: 1. elem sötét színe, közepes színe, világos színe; 2. elem sötét színe, közepes színe, világos színe; és így tovább.

Térjünk rá a palettakészítésre. A QB-nek van egy **PALETTE** utasítása, csak ennek van egy kis hibája. Lassú. Valami rejtélyes okból egy komplett paletta felépítése jó néhány másodpercbe telik. Ezért a dörzsölt QB-programozó inkább egy egészen más módszert használ, egy kész rutint, amit egyből be is másolhatunk a programba:

```
DEFINT A-Z
SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, b
END SUB
```

Ez a százezer éves kis rutin szempillantás alatt elvégzi egy színkód átállítását. Bemenő paraméterei: **p**, vagyis a színkód 0-tól 255-ig; **r**, vagyis a piros *(red)* színkód 0-tól (legsötétebb) 63-ig (legvilágosabb); **g**, vagyis a zöld *(green)* színkód ugyanígy; és **b**, vagyis a kék *(blue)* színkód szintén ugyanígy.

Ha úgyis új színkódokat kapnak az elemeink színei, akkor át is tudjuk rendezni őket. A **szin** tömb nyugdíjba mehet, a *LISZTOJ* szerinti első elem színkódja a 36-os színtartomány első színe (azazhogy színhármasa) lesz, a második elemé a második szín, és így tovább. Nézzük tehát át az elemeket a *LISZTOJ* sorrendjében, és válogassuk ki a színeiket.

Ahhoz, hogy ezt megtehessük, kell egy színválogató eszköz, elvégre senki se képes képzeletben színeket válogatni. Csináljunk egyet. Írjunk egy rutint, ami segít nekünk a színkeresésben. Legyen a neve **szinek** és csináljon annyit, hogy átkapcsol a grafikus képernyőre és a 0-s színt, vagyis a hátteret egy kívánságunk szerinti színre festi.

```
DEFINT A-Z
SUB szinkeres
SCREEN 13
paletta 0, 32, 32, 32
END SUB
```

Ennyi az egész, nincs más dolgunk, mint az *Immediate* ablakból meghívni a rutint, aztán a három 32 helyére tetszés szerint írogathatunk számokat.

Az elemek kapjanak más, új színeket. Ha csoportosítani akarjuk az elemeket, akkor a négyzet alakúak csoportjába kerül az **O** és a **K,** az egyenesek közé a **D, H** és **I,** a derékszögűekhez az **L, J** és **V,** a cikcakkosakhoz a **Z** és **S,** és végül egyedül marad a **T.**

Kezdjük a két háromtagú csoporttal. Az egyenesek legyenek semleges színűek: fehér, szürke, másik szürke. A derékszögűek kapjanak meleg színeket: piros, sárga, narancs. Jó, akkor a hideg színek jók lesznek a cikcakkosakhoz: zöld és kék, a négyzet alakúak pedig legyenek lila és másmilyen lila, a **T** pedig barna.

Elméletben készen vagyunk, lássuk a gyakorlatot. Az egyenesek közül mondjuk a **H** legyen fehér, mert ha az is szürke, akkor a hasonló hosszúság miatt könynyen összetéveszthető a másik szürkével. Tehát a **H** fehér, ez úgy lesz, hogy mindhárom színösszetevő maximális, vagyis 63. Legyen mondjuk a **D** világos, az **I** pedig sötétszürke. Csináljunk világosszürkét. Szürkét úgy kapunk, hogy mindhárom színösszetevő egyenlő marad, minél kisebbek, annál sötétebb a szürke. Mondjuk nézzük meg világosszürkének a 40, 40, 40-et, sötétnek pedig a 20, 20, 20-at.

Nézzük a derékszögeket. A piros legyen teljesen élénkpiros (63, 0, 0), ez lehet az **L,** a **J** pedig sárga (63, 63, 0). A narancs úgy keletkezik, hogy a zöld fele a pirosnak: 63, 31, 0.

A cikcakkosak zöld és kék, ez 0, 63, 0 és 0, 0, 63.

A négyzet alakúak lila és másmilyen lila. Lilát úgy kapunk, hogy ugyanannyi pirosat és kéket használunk, zöldet pedig egyáltalán nem. Az egyik lehet 63, 0, 63, a másik pedig mondjuk 31, 0, 31, vagyis sötétlila.

Végül pedig a **T** barna lesz; ezt úgy kapjuk meg, mint a narancssárgát, de sötétebben. Mondjuk 31, 15, 0.

Oké, eddig minden jó, tegyük be a palettába… jaj, dehogy jó. Egyáltalán nem jó! Arról volt szó, hogy minden színnek lesz egy sötétebb, egy normál és egy világosabb változata. Mi elméletileg a normál változatokat kerestük ki, és a színek jelentős hányada a lehető legvilágosabbra sikerült. Ugyanis a legpirosabb piros egyúttal a legvilágosabb piros (ami még tisztán piros, nem hajlik sárgába, lilába vagy rózsaszínbe). Akkor legyen ez a legvilágosabb piros a piros színhármasunk *világos* színe, a *normál*t pedig mondjuk öttel csökkentve érjük el. Tehát nem 63, hanem csak 58. Akkor mindenütt, ahol a színeinkben 63 szerepel, írjunk 58-at, a többi számot viszont nem kell csökkenteni.

Így lesz jó, tekintsük át a színeinket.

elem		szín	R	G	B
	L		58	0	0
	I		20	20	20
	S		0	58	0
	Z		0	0	58
	T		31	15	0
	O		31	0	31
	J		58	58	0
	K		58	0	58
	D		40	40	40
	H		58	58	58
	V		58	31	0

Most vegyük mindegyik színben azokat a számokat, amik nem nullák, és csökkentsük őket öttel. Ezek lesznek a sötétebb színek. A világosabbakat úgy kapjuk meg, hogy öttel növeljük a számokat. Az **L** színéből ez a hármas lesz:



A programozó összeráncolja a homlokát. Elég nagy a különbség ahhoz, hogy térbeli hatást lehessen elérni a párosításukkal? Próbáljuk ki. Legjobb lesz az **inic** végén kialakítani a palettát. A számtartományunk ugye 16-tól kezdődik, ez az első színhármas, tehát a három piros közül a legsötétebb a 16-os kódú szín. Csináljuk meg őket.

paletta 16, 53, 0, 0
paletta 17, 58, 0, 0
paletta 18, 63, 0, 0

Próbáljuk is ki. Rajzoljunk velük térbeli alakzatot a képernyőre.

LINE (0, 0)-STEP(10, 10), 18, B
LINE (2, 2)-STEP(10, 10), 16, B
LINE (1, 1)-STEP(10, 10), 17, BF
STOP

Valami nem az igazi. Lett egy piros kockánk a képernyő sarkában, de ugyanakkor ott van a képen az akna teljes rajza, mindenestül. Mi rajzolja ki? A program az **inic**cel kezdődik, az pedig nem tesz a képernyőre semmit. Igaz, meghívja a **racsinic**et. Nézzük csak… tényleg, ez egy **aknarajz**zal végződik. Ezt vegyük innen ki, az akna úgyis ki lesz majd rajzolva annak rendje-módja szerint. És nézzük a piros kockánkat.

Hát ez bizony híjával van a térbeliségnek. Kevés az az öt különbség. Növekedjenek a számok tízesével, 43-53-63.

Ez már jó lesz. Az első színhármasunk megvan, és ezen az elven a többbit is ki tudjuk dolgozni. A normál színekhez 58 helyett 53-at írunk, a sötétebbeknél minden számot tízzel csökkentünk, a világosabbaknál tízzel növelünk.

De azért ezt mégse írjuk harminchat darab **paletta** utasítással. Elég lesz tizenkettő, a másik két garnitúrát ciklus is hozzáteheti. Töröljük ki a **paletta** utasításokat és a rajzolást az **inic**ből, a helyére kerüljön a ciklus:

FOR sz = 0 TO 11
FOR f = 0 TO 2

A külső ciklus végigmegy a színeken, a belső a fényességeken. Ha a színek így sorakoznak egymás után a palettában:

16: sötét piros
17: közepes piros
18: világos piros
19: sötét sötétszürke
20: közepes sötétszürke
21: világos sötétszürke

stb., akkor egy adott színhármas első tagjának kódját úgy kapjuk meg, hogy a színhármas sorszámát beszorozzuk hárommal és hozzáadunk 16-ot. Ehhez még hozzáadva a fényességet megkapjuk az adott színt. Ha egy **paletta** utasításban mondjuk 31-es szám áll egy színösszetevőnél, akkor az igazából csak akkor 31, ha **f = 1**, mert ha **f = 0**, akkor le kell belőle vonnunk 10-et, ha viszont **f = 2**, akkor hozzá kell adnunk 10-et. Hogy lehet ezt minél egyszerűbben? És ha azt monda-

nánk, hogy **31 + f * 10**? **F** pedig ne 0-tól 2-ig fusson, hanem –1-től 1-ig, és majd nem 16-ot adunk hozzá, hanem 17-et. Rakjuk össze a dolgokat és nézzük meg, hogy működnek-e.

```
FOR sz = 0 TO 11
FOR f = -1 TO 1
paletta sz * 3 + f + 17, 53 + f * 10, 0, 0
LINE (sz * 10, f * 10 + 10)-STEP(9, 9), sz * 3 + f + 17, BF
NEXT: NEXT
STOP
```

Így fest a kód némi agyalás után. Kellett bele valami, ami rajzol is a színekkel, hogy lássuk, mit műveltünk. Három különböző árnyalatú piros csíkot kapunk… igen ám, de mi nem akarjuk mind a tizenkét színt pirosra festeni, eszünk ágában sincsen. Tegyük bele a második színt is:

```
paletta sz * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
```

A színkódot azért hagytam változatlanul, mert nem tudtam, mit írjak oda. Persze, mi egyáltalán nem akarunk ciklussal végigmenni a színeken, nekünk minden egyes színt külön kell beállítanunk. Dobjuk csak ki onnan azt az **sz** ciklust, és az első **paletta** utasításban 0 álljon **sz** helyén, a másodikban 1. Kirajzolni így már csak úgy fogja, ha minden **paletta** alá bemásoljuk a **LINE** utasítást, átírva benne **sz** mindkét előfordulását a fölötte levő sor első számára. No jó, csináljuk meg a többi színt is. Az **S** elem színe:

```
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
```

Ezen a ponton viszont kis gubanc van. A zöld oszlop felső kockája szürkével jelenik meg. Vajon a rajz rossz vagy a palettakészítés? Ellenőrizzük máshogyan is.

```
FOR p = 16 TO 51
LINE (p * 2, 40)-STEP(1, 5), p, BF
NEXT
```

Ez is így rajzolja ki, a szürke rész szélesebb, mint a zöld, több szín van benne. Tehát hibás a palettakészítés.

A programozó természeténél fogva gyanakvó, bizalmatlan ember. Nem bízik a gépben. A **STOP** elé beírja: **a$ = INPUT$(1)**, és így is elindítja a programot.

Láss csudát, a harmadik oszlopban három tökéletes zöld sorakozik, és a pótlólag kirajzolt csíkban is ott van a zöld. A programozó megnyom egy gombot, a **STOP** hatására visszatér a QB képernyője, a programozó megnyomja az *F4*-et és megszemléli a képet, amin a zöld helyén már megint szürke van. A programozó elkönyveli, hogy a gép időnként ritka hülye tud lenni, és nem foglalkozik tovább a dologgal. Hiszen ha a zöld csak akkor változik szürkévé, ha a program megáll egy **STOP**-nál, akkor nincs probléma. A pótlólag betett ciklus nem kell, mehetünk a többi színhez.

```
LINE (3 * 10, f * 10 + 10)-STEP(9, 9), 3 * 3 + f + 17, BF
paletta 4 * 3 + f + 17, 31 + f * 10, 15 + f * 10, 0
LINE (4 * 10, f * 10 + 10)-STEP(9, 9), 4 * 3 + f + 17, BF
```

```
paletta 5 * 3 + f + 17, 31 + f * 10, 0, 31 + f * 10
LINE (5 * 10, f * 10 + 10)-STEP(9, 9), 5 * 3 + f + 17, BF
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
LINE (6 * 10, f * 10 + 10)-STEP(9, 9), 6 * 3 + f + 17, BF
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
LINE (7 * 10, f * 10 + 10)-STEP(9, 9), 7 * 3 + f + 17, BF
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
LINE (8 * 10, f * 10 + 10)-STEP(9, 9), 8 * 3 + f + 17, BF
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
LINE (9 * 10, f * 10 + 10)-STEP(9, 9), 9 * 3 + f + 17, BF
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0
LINE (10 * 10, f * 10 + 10)-STEP(9, 9), 10 * 3 + f + 17, BF
```

Fél pillanat, arról volt szó, hogy 11-ig megyünk a színekkel, elvégre 0-tól kezdtük és 12-en vannak, de mi az utolsó szín? A programozó visszaszalad a fentebbi színlistába és megszámolja az elemeket. Tizenegyen vannak. Akkor hogy-hogy tizenkettő? Kimaradt valamelyik? A programozó visszarohan a főprogramba és megnézi a **DATA** utasításokat. Persze hogy tizenkettő, de az extrával együtt!

Rezerváljunk egy színhármast az extrának is? Hát lehet éppen. No de ki mondta, hogy nekünk mindig csak egyféle extránk lesz a programban? Több is lehet. A különféle extrák célszerűen más-más kinézetűek lesznek, hát akkor hány színhármast tartsunk fönn nekik? Legjobb, ha egyet sem, az extrák színeit majd máshol tároljuk. Így a színeink száma lecsökken háromszor tizenegyre, az 33, nem pedig 36, vagyis az első színtartomány 16-tól 48-ig megy majd, a zöldesített 49-től 82-ig, a pirosított pedig 83-tól 114-ig.

Csináljuk meg ezeket is. Ehhez persze némi morfondírozás fog kelleni.

Zölddel ugye úgy keverünk egy színt, hogy valamennyit hozzáadunk a zöld színösszetevőjéhez. Igen ám, de egyes színeinknél nem tudjuk megtenni, mert a zöld színösszetevőjük már így is magas. A zöld, a sárga és a fehér ilyen. A pirossal való keverésnél se járunk jobban, mert az a színösszetevő pedig a piros, a világoslila, a fehér és a narancs esetén magas. Máshogyan kell nekifognunk.

Nézzük a fehér színt. Mi lesz ebből, ha zölddel keverjük? Hát ugye zöld. Hogyan csinálhatunk a fehérből zöldet? Úgy, hogy lecsökkentjük a piros és a kék színösszetevőjét. Akkor próbáljuk azt, hogy ne a zöldet növeljük, hanem a másik kettőt csökkentsük? Nem jó, mert például ha ezt a pirossal vagy a kékkel próbáljuk megcsinálni, akkor egyszerűen sötétebb pirosat és kéket kapunk, márpedig mi sárgába vagy ciánkékbe hajló színt szeretnénk. Tehát az egyik színnél a zöldet kell növelni, a másiknál a pirosat és a kéket csökkenteni? Ez elég macerásnak tűnik. Az is, de csak egyszer kell megcsinálni.

Kezdjük a zölddel. Sorakoztassuk föl a színeink normál fényességének összetevőit, és nézzük meg, melyikkel mit csináljunk.

R	G	B	teendő
53	0	0	G🡅
20	20	20	G🡅
0	53	0	G🡇
0	0	53	G🡅
31	15	0	G🡅
31	0	31	G🡅
53	53	0	R🡇
53	0	53	G🡅
40	40	40	G🡅
53	53	53	RB🡇
53	31	0	G🡅

A táblázat végül is eléggé egyszerűen állt elő. Mindenhol G🡅-t írtam, ahol a G kisebb volt 53-nál. Ha 53 volt, akkor lefelé vittem a másik (két) színösszetevőt, ha lehetett. Ahol nem lehetett, ott pedig azt írtam, hogy G🡅, mert az egészen világos zöldet sötétebb zölddel keverve logikusan sötétebb zöldet kell kapnunk.

Oké, most vigyük át a gyakorlatba. Gyomláljuk ki ezt a töméntelen **LINE**-t az **inic**ből, mert nem lehet látni tőlük, és másoljuk le az összes **paletta** utasítást még egyszer. Most tegyük be mindegyikbe, a színkód elejére, hogy **33 +**. Ezzel átmásoltuk színeinket a zöldítendő területre. Most zöldítsünk. Legyen **z**-ben a változtatás mértéke, ez legyen mindig ugyanannyi, akár a zöld színösszetevőt változtatjuk, akár másikat. Az első két sorban a zöld színösszetevőhöz, vagyis az utolsó előtti számhoz adjunk hozzá **z**-t, a harmadik sorban vonjuk le, a következő háromban megint adjuk hozzá, a következőben viszont a pirosból vonjuk le, kétszer megint a zöldhöz adjuk hozzá, az utolsó előttiben a pirosból és a kékből is vonjuk le, az utolsóban megint a zöldhöz adjuk hozzá. Ez mindenütt **+ z** vagy **- z** hozzáírogatását jelenti, legfeljebb ahol nulla áll, ott egyszerűen kicseréljük **z**-re.

Persze ahhoz, hogy ez működjön, kell egy **z** is; egyelőre menjünk vissza az elejére és írjuk be, hogy **z = 20**. Majd aztán meglátjuk, hogy mennyi lesz megfelelő.

Tegyük láthatóvá munkánkat; vegyük elő a korábban már használt

```
FOR p = 16 TO 51
LINE (p * 2, 40)-STEP(1, 5), p, BF
NEXT
```

ciklust, írjuk be az **f** ciklus mögé, csak most ne 51-ig fusson, hanem 33-mal tovább, 84-ig.



Hát tagadhatatlanul van valami zöldes hatás a jobb felén, nem mindenütt ugyan, de nem az volt a cél, hogy zöld legyen, hanem hogy zöldebb, mint a kiindulás. Így nehéz összehasonlítani; jobb, ha menet közben látjuk a dolgot. Pihen-

jünk egy kicsit és írjuk át a programot úgy, hogy ezeket a színeket használja. Aztán ha jó, akkor megcsináljuk a piros színtartományt is.

```
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB kockarajz (x%, y%, szin%)
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk, 5
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk, 3
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk, 14
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk, 2
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk, 4
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk, 6
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk, 1
DATA k, k, k, k, 15
DATA kjk, klk, kjk, klk, 11
DATA kbkjjk, kfkllk, kbkjjk, kfkllk, 7
DATA kfkljk, kjkblk, kbkjlk, kbkjfk, 13
DATA k, k, k, k, 255

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
```

```
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e)
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE kockatorles ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 100
akna(aknax, y).szin = 100
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 100
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: READ szin(e): NEXT
```

```
RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 100
LINE (px, py)-STEP(3, 3), 12, BF
LINE (px + 5, py)-STEP(3, 3), 12, BF
LINE (px + 2, py + 5)-STEP(3, 3), 12, BF
LINE (px + 7, py + 5)-STEP(2, 3), 12, BF
LINE (px, py + 5)-STEP(0, 3), 12
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
CASE ELSE
```

```
LINE (px + 1, py + 1)-STEP(7, 7), szin, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 7, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
```

```
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT

aknarajz
END SUB

SUB racsrajz (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
FOR rx = px TO px + 9
FOR ry = py TO py + 9
IF (rx + ry) / 2 = (rx + ry) \ 2 THEN PSET (rx, ry), akna(x, y).tolt * 2 + 8
NEXT: NEXT
END SUB

SUB ragadas (e, i, ex, ey)
IF szin(e) = 255 THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
```

```
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
kockatorles x, 1
NEXT
aknarajz
END SUB
```

Kifestő

Mi jön most? Fessük át az elemeinket az új színekre. Ez nem lesz nehéz, mert a színeiket pillanatnyilag egy tömb adja meg, amit könnyedén átváltoztathatunk függvénnyé. Csak ki kell törölnünk a deklarációját a főprogramból és a feltöltését (a **READ** utasítást) az **inic** rutinból; persze emiatt el kell távolítanunk a számokat a **DATA** sorok végéről.

A függvénynek annyi lesz a dolga, hogy az elem sorszámát átalakítsa a színévé. Legjobb, ha a normál fényességű színné alakítja, mert egyelőre úgyis csak azzal rajzolunk.

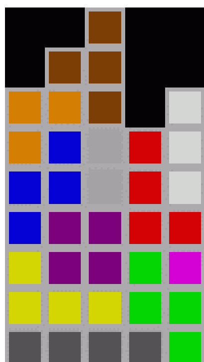
```
DEFINT A-Z
FUNCTION szin (e)
szin = e * 3 + 17
END FUNCTION
```

Ennyi az egész, nézzük meg, mit alkottunk. Persze az **inic** végén még megtorpan, tegyük az egész rajzolgatást és a megállást **'**-ok mögé.



Munkánk, finoman kifejezve, kissé tökéletlen. Összegyűjtöttem egy példányt az összes elemből. Nem azokat a színeket viselik, amiket kellene.

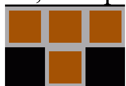
Vajon miért? Ó, persze. A **szin** függvénybe írt képletet az **inic** palettabeállító utasításaiból vettük, ahol az **e** helyén 0, 1, 2 stb. állt. Ez a függvény tényleg csak akkor működik helyesen, ha az elemeket nullától számozzuk, de mi eddig egytől számoztunk. Így mindegyik elem az utána következő színét kapta, az utolsó pedig a zöld színtartomány első színét. Menjünk vissza **szin**be és csökkentsük a visszaadott értéket 3-mal. Legegyszerűbb, ha a 17 helyére 14-et írunk.



Mindjárt jobb. Az egyenes elemek háromféle szürkék, a derékszögűek pirosak, sárgák és narancsok, a cikcakkok zöldek és kékek, a kockák világos- és sötétlilák, a **T** elem pedig barna. Eddig tehát megvagyunk.

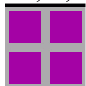
A régi programozómondás szerint a puding próbája az, hogy megeszik. *Most* lehet megnézni, hogy jók-e a színeink, működés közben, amikor játszunk az elemekkel. A programozó kicsit kritikusan néz a **T**-re, az **I**-re és az **O**-ra, túl sötétnek tartja őket. A **D**-nek nem előnyös a szürke keret, szinte egybefolyik vele, de ez a keret úgyse marad ott sokáig, azért csináltunk sötétebb és világosabb színváltozatokat, hogy azokkal keretezzük be őket. Talán a **V** is lehetne egy kicsit világosabb, gondolja a programozó, és állítgat kicsit a színeken.

A **T** a *LISZTOJ* ötödik eleme, tehát nullától számozva 4-es; az **inic** rutin **paletta** utasításaiban a 4-est kell megkeresni. 31, 15, 0 a kódok. Legyen mondjuk 40, 20, 0. A programozó átírja és kijavítja a zöld színtartományt is hasonlóképpen.



Már jobb, bólint a programozó, és elteszi magában azt az információt, amit a **T** elemre várva figyelt meg: a **K** elemek egy része lila, más része sötét narancssárga, és bomba nincsen.

Lássuk a következő elemet, az **O**-t. Színkódja 31, 0, 31, hát akkor ez is legyen 40, 0, 40.



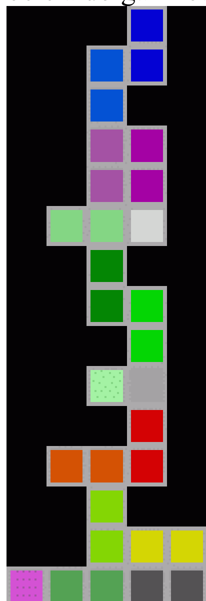
Ez is jó. Az **I**-t esetleg hagyjuk egyelőre békén, nem kell, hogy túlságosan hasonlítson a **D**-re. A **V** is várhat. Nézzük meg inkább a zöldítést.

Morfondírozzunk. Mi kell ahhoz, hogy a zöld rács mögé tett elemeket zölden, vagyis rendes színkódjuknál 33-mal magasabb színben lássuk? A rács ki-

rajzolását a **racsrajz** rutinra bíztuk… de a rács mögötti kocka képe most már teljesen ugyanolyan, mint a rendes kockáé, csak a színek mások – ezért jobb lesz a **kockarajz**zal intéztetni, csak a színkódokat változtassuk meg. Mondjuk kapjon a **kockarajz** egy új paramétert, ami megmondja, hogy eltérő színben kell-e kirajzolnia. Ha ez 0, akkor rendesen, ha 1, akkor zöldítve, ha pedig 2, akkor pirosítva, vagyis az **akna** tömb **tolt** mezőjének értékeit használjuk itt is, ez még jól fog jönni. Tehát legyen az új paraméter **t**, és ha az értéke 1, akkor adjunk 33-at a színkódokhoz. Ezt a **CASE ELSE** utáni sorban kell megtennünk, ahol **szin**t kijavítjuk **szin + t * 33**-ra. Lássuk!

Még nem látjuk, először is **Argument-count mismatch**, mondja a főprogramban a **kockarajz** deklarációjára; azért teszi, mert a QB által automatikusan odaírt rutindeklaráció nem egyezik a valóságossal. Töröljük ki ezt a sort, majd mentéskor visszaírja az új változatot. Újabb indítási kísérletre is ugyanezt kapjuk, csak most a **kockarajz** hívásánál. Csakugyan, meg kell adnunk **t**-t is. **Elemrajz**ban nincs szükség semmilyen zöldítésre-pirosításra, tehát **t** nulla lesz, viszont **aknarajz**ban már igenis szükség van rá, tehát ott **akna(x, y).tolt** lesz az értéke.

Nemigen van változás, persze, mindenekelőtt a **racsrajz**zal meg kell beszélnünk, hogy ne sakktáblázza össze a kockáinkat. Egyelőre iktassuk ki, ne kavarjon bele a dolgainkba. Tegyünk az elejére egy **EXIT SUB**-ot.



Ilyenek a színeink. A képhez magyarázat is kell, mert enélkül nem érthető. Felül egy kék **Z** elem van, alatta lila **O,** ez alatt fekszik egy fehér **H,** majd egy zöld **S,** aztán egy világosszürke **D,** egy piros **L,** egy sárga **J,** legalul pedig egy lila **K** és egy sötétszürke **I.** A **T** és a **V** elem nem fért a képre, ezek külön vannak jobbra. Egyelőre fogjuk rá, hogy jó, majd később még teszteljük és finomítunk rajta, ha kell.



Most inkább szerezzük vissza a bombánkat. A bomba ugyanis nyomtalanul elveszett a programból, helyette kétféle színű **K** elem van. Csakugyan, hiszen a bombát eddig az különböztette meg a rendes **K** elemtől, hogy 255-ös színkódja volt, azt viszont most elvettük tőle. Nem is tudjuk visszaadni, hála új módszerünknek, a színkódokat most már kiszámoljuk, nem tömbből vesszük, mást kell kitalálnunk. A **DATA** sorban most már csak a ceruzakódok állnak, semmi egyéb. Hát akkor különböztessük meg őket a ceruzakóddal. Az elem csupa **"k"** kódból áll. Mi lenne, ha az extra jelzéséül ezt lecserélnénk **"x"**-re? Akkor a ceruzakód értelmezésénél kell lekezelnünk az új helyzetet, nem a színkódnál.

Az **elemrajz**ot ki kell bővítenünk egy **CASE "x"** esettel. Mi történjen itt? Hozzuk ide **kockarajz**ból a golyórajzolást vagy hívjuk inkább meg? Inkább hívjuk

meg, logikusabb, ha az elemek egy kockányi részét kirajzoló rutinban van az egykockányi méretű extra kirajzolása is. Tehát **kockarajz x, y, 255, 0**.

Hát megjelenni megjelenik a bomba, csak nem törlődik mozdítás után és nem fejti ki hatását. Debugoljunk.

Először is legjobb lesz átírni az **ujelem**et úgy, hogy kicsit gyakrabban adjon bombát; ne mindig azt, mert akkor nem lesz mit eltüntetni vele, mondjuk legyen **e = INT(RND * 3 + 10)** és **'** mögé az eredeti szöveg. A **mozgas**ba, a **DO** után pedig tegyünk egy **IF e = 12 THEN STOP**-ot, innentől kövessük lépésenként az eseményeket. A végrehajtás hamarosan átlép az **INKEY$**-on, mi pedig rájövünk, hogy nem tudjuk lenyomni semelyik gombot, hogy a program elágazhasson; sebaj, ott az *Immediate* ablak, beírjuk, hogy **a$ = CHR$(0) + "P"**, és mehetünk tovább. Nemsokára elérjük az **elemtorles**t, ahol a homlokunkra üthetünk. Persze, ez végigmegy a ceruzakódon és akkor töröl, ha **k** betűt talál, no de itt csak **x** van. Tegyük csak oda, hogy **CASE "k", "x"**!

Eddig oké, a bombánk most már letörlődik (ha kivesszük azt az **IF e = 12 THEN STOP**-ot, mert különben kínszenvedés mozgatni), csak ahelyett, hogy robbantana, keresztülmegy a kockákon. Nézzük csak át a programot, keressünk a ceruzakódot értelmező részeket. Persze, **mehet**ben és **ragadas**ban is van ilyen. Utóbbira elvileg nem lesz szükség, a bombát külön kezeli, de **mehet**be tegyük be az **x**-et is a **k** mellé.

Oké, a bomba most már fönnakad a kockákon, csak nem robbant. Miért is nem? Hát persze, mert a **ragadas** felelős az **extra** meghívásáért, de ezt akkor teszi, ha a színkód 255. Szerencsére a **ragadas** akkor is fel tudja ismerni, hogy extrával van dolga, ha a színkód nem segít neki: az extra ceruzakódja **"x"**. Tehát **IF raktar(e, i) = "x" THEN** kerül az első sorba.

A bomba most már működik.

Csináljuk meg most már a rácsot is rendesen, mert maga a rács egyáltalán nem látszik, csak a mögé tett kocka változtat színt. A **racsrajz** szövegéből gyakorlatilag mindent kidobálhatunk, csak a **px**-nek és **py**-nak való értékadás marad meg. A rajzolás nagyon egyszerű lesz, csak befestjük az egész négyzetet – de milyen színnel? A rács színe ugyanis hiányzik a palettáról. Csináljuk meg. Az 1–15 színkódok most már nem kellenek semmi különösre, használjuk ezeket. Az 1-es színkód legyen a zöld rács. Az **inic**ben 30-at adogattunk hozzá és vonogattunk le, legyen a rácsunk színe 30-as zöld, úgy értem, 0, 30, 0. Tegyük be az **inic**be, a **racsrajz**ba pedig rakjuk be: **LINE (px, py)-STEP(9, 9), 1, BF**.

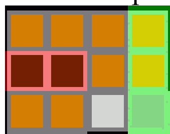
Egész jól néz ki, csak eltűnnek a kockák. Persze, rájuk rajzoljuk a rácsot. Tegyük be a **racsrajz**ba, hogy **IF akna(x, y).szin = 0 THEN**.

Minden szép és jó, csak ha bombát robbantunk a rács előtt, amikor alatta az akna feneke van, akkor fekete kocka marad. Nézzünk be az **extrá**ba. Ha az akna fenekén robban a bomba, akkor csak egy **kockatorles** van beírva. Tegyünk mögé egy **racsrajz ex, ey**-t is.

Ez eddig rendben van, csináljuk meg a piros rácsot is. Legyen a 2-es paletta-szín az övé, és legyen 30, 0, 0. Ha betettük az **inic**be, akkor csak annyi a feladat, hogy a **racsrajz**ot értesítsük a változásról. A **LINE** utasításban a színkódot egyszerűen átírhatjuk **akna(x, y).tolt**re, és kész.

Most már minden szép és jó – az egyetlen gond a kockák szürke kereteivel van, eléggé illúzióromboló, hogy nem zöldülnek meg. Zöldítsük meg őket. Kell két palettaszín, egy zöldített és egy pirosított. De kell egy sima is, miért legyen pont a 7-es a szürke, sorakozzanak szépen rendben. Tehát a 3-as lesz a rendes szürke, a 4-es a zöldített és az 5-ös a pirosított. Legyen a rendes szürke 30, 30, 30, akkor a színezett változatokban a megfelelő színösszetevőt 60-ra állíthatjuk.

Akkor használjuk föl az új színeinket. A **kockarajz** végén, az utolsó **IF** utasításban van a szürke keretünk, itt a 7-es színkód helyett **3 + t**-t írhatunk, és meglesz a zöldített-pirosított szürkénk.



Így néznek ki az új keretek, kicsit sötétebbek a régieknél, de szépen pirosodnak és zöldülnek. Állítsuk most már vissza **ujelem** régi állapotát és játszadozzunk így el a programmal.

```
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED rcs
```

```
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB
```

```
SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 100
akna(aknax, y).szin = 100
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 100
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
```

```
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 100
LINE (px, py)-STEP(3, 3), 12, BF
LINE (px + 5, py)-STEP(3, 3), 12, BF
LINE (px + 2, py + 5)-STEP(3, 3), 12, BF
LINE (px + 7, py + 5)-STEP(2, 3), 12, BF
LINE (px, py + 5)-STEP(0, 3), 12
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 12
PAINT (px + kocka / 2, py + kocka / 2), 12
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT
FUNCTION
END SELECT
NEXT
END FUNCTION
```

```
SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
```

```
NEXT: NEXT
FOR x = 2 TO 12
akna(x, aknay - 3).tolt = 2
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
```

```
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
kockatorles x, 1
NEXT
aknarajz
END SUB
```

Piroska

Csináljuk meg most már a pirosított színváltozatokat is, csúnyák a kockák, amikor a piros rácsra kerülnek. Jobban tudunk kísérletezni velük, ha a **racsinic** rutin **CASE 1** részénél a pirosító ciklust átírjuk ilyenformán:

```
FOR y = 10 TO 19
FOR x = 2 TO 12
akna(x, y).tolt = 2
NEXT: NEXT
```

Így szép nagy piros felületet kapunk, lesz hol próbálgatni a színeket.

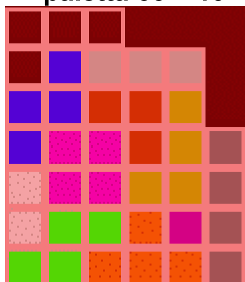
Lássuk a pirosítást. Az **inic**ben másoljuk le az első garnitúra **paletta** utasítást, és a másolaton mindenhova írjuk be, hogy **66 +**, ez lesz a piros színtartomány. Csináljunk megint egy olyan táblázatot, amilyet a zöldítésnél.

R	G	B	teendő
53	0	0	R🡇
20	20	20	R🡅
0	53	0	R🡅
0	0	53	R🡅
31	15	0	R🡅
31	0	31	R🡅
53	53	0	G🡇
53	0	53	B🡇
40	40	40	R🡅
53	53	53	GB🡇
53	31	0	R🡇

S most vigyük át a számokat a **paletta** utasítások harmadik csoportjára. Legyen a hozzáadandó változónk neve **p**, az értéke pedig legyen szintén 20.

```
p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
```

```
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
```



Ilyenek az elemeink pirosított állapotban. Igazából nem is rossz.

Az új palettára való átállás azonban még nincs készen: a golyó és a fal még a régi színeket használja. Célszerű lesz új színeket adni nekik is.

1-től 5-ig és 16-tól 114-ig már lefoglaltuk a palettát. Legyen a fal piros színe 6-os, a szürke 7-es, de kicsit változtassuk meg őket. A piros ne piros legyen, hanem terrakotta, mondjuk 60, 20, 20-as, a szürke pedig kicsit sötétebb és kékesbe hajló, esetleg 20, 20, 25. Írjuk be a **paletta** utasításokat, aztán nézzük **kockarajz**ot. A 100-as színkód jelzi a falat. Itt egyszerű a dolgunk, csak a 12-es színt kell kicserélni 6-osra, a 7-es változatlan marad.



Nem is rossz. Oké, egyelőre maradjunk ennél. Lássuk a golyót. Mi lenne, ha 6-os színnel rajzolnánk? Nem az igazi. Legyen inkább 8-as számmal egy 63, 0, 0-s tűzpirosunk, ezzel rajzoljuk meg.

```
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, b%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB zuditas (y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
END TYPE

CONST elemek = 12
```

```
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
```

```
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 100
akna(aknax, y).szin = 100
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 100
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
```

```
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 100
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
```

```
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin THEN mehet = SGN(x - ex) + 3 - (x - ex = 2): EXIT
FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
```

```
OUT &H3C9, b
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR y = 10 TO 19
FOR x = 2 TO 12
akna(x, y).tolt = 2
NEXT: NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
```

```
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y)
FOR yy = y TO 2 STEP -1
FOR x = 2 TO aknax - 1
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
NEXT: NEXT
FOR x = 2 TO aknax - 1
akna(x, 1).szin = 0
kockatorles x, 1
NEXT
aknarajz
END SUB
```

Kőművesmunka

Újabb gondolat merült föl. Mi történne, ha a remekbe szabott falainkat nemcsak az akna két oldalán és az alján használnánk, hanem bent az aknában is? Az akna belsejében itt-ott falak húzódnának, amik akadályozzák az elemek mozgását.

Mi sem egyszerűbb ennél, tegyünk falakat az aknába. Ott a **racsinic**, az elhelyez néhány 100-as kódot a **szin** mezőbe, és kész.

```
17 FOR y = aknay - 5 TO aknay - 1
akna(18, y).szin = 100
NEXT
```

Csakugyan megjelenik egy függőleges faldarab az aknában, probléma egy szál se.

Mindaddig, amíg be nem töltünk egy sort és ki nem derül (nem mintha magunktól nem tudtuk volna), hogy a fal is megy lefelé a kockákkal együtt. De egy téglafalnak akkor van igazán értelme, ha kikerülhetetlen, eltüntethetetlen akadályként áll a játékos útjában, amivel nem lehet mást tenni, mint tudomásul venni, hogy ott van és eszerint kialakítani a stratégiánkat.

Ez azért már nem olyan egyszerű dolog: morfondírozást igényel. Odáig rendben is van, hogy ha betöltünk egy sort, akkor a fal nem megy lejjebb, erről külön szólunk a **zuditas** rutinnak: amelyik kockában fal van, azt ne vigye lejjebb. Esetleg próbáljuk ki. De a fal színkódja ne 100 legyen ezentúl, mert az egy palettaszín a három tartományunkból, legyen inkább olyan kód, ami nem fordulhat elő rendes színként. Legyen 256. Ha a **zuditas** ezt a számot találja egy kockában, akkor ne vigye lejjebb.

IF akna(x, yy - 1).szin < 256 THEN akna(x, yy).szin = akna(x, yy - 1).szin

Semmi nem változott, a fal ugyanúgy eltűnik. Vegyük ki a **- 1**-et a feltételből.

Most már jó, a fal nem tűnik el. Azért a biztonság kedvéért tegyünk az akna közepére is egy vízszintes faldarabot:

20 FOR x = 17 TO 25
akna(x, 10).szin = 256
NEXT

Ezt tettem a **racsinic**be, próbáljuk ki így is.

Hát igen, mégse tökéletes a dolog. Az első sor betöltésekor a vízszintes faldarabunk alatt megjelent egy másik vízszintes faldarab.

Kolumbusz tojása. Kezeljük a falat teljesen külön! Legyen az **akna** tömbnek egy **fal** mezője is, aminek 1-es értéke jelzi, hogy fal van abban a mezőben. A **zuditas** csak a **szin** mezőre figyel, nem véletlen, hogy a rácsaink se mozdulnak, akárhogyan töltögetjük be a sorokat.

Tehát tegyük be a **TYPE** utasításba ezt a mezőt is, aztán a **racsinic**ben írjuk át, hogy ne 256-osokat rakosgasson a **szin**be, hanem 1-eseket a **fal**ba.

Persze ettől meg eltűnik a falunk. Oldjuk meg, hogy mégis kirajzolódjon: az **aknarajz** figyelje, hogy van-e fal egy mezőben, és ha igen, rajzoltassa ki.

5 IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE

és ebbe az **ELSE**-be tegyük bele a másik **IF** utasítást.

Eddig jó, a fal már megjelenik, csak az elemek mennek rajta keresztül. Persze, mert a **mehet** üres mezőket talál ott, ahol fal van. Írjuk bele a **CASE "k", "x"** alatti **IF**-be, hogy **OR akna(x, y).fal = 1**.

Oké, az elemek már nem mennek át a falon, sőt, ha betöltünk egy sort, a fal mozdulatlan marad – csakhogy ez a sor csak olyan lehet, amiben nincsen fal. Azokat a sorokat, amikben van fal, egyáltalán nem lehet betölteni. Nézzük meg a **sorok** rutint. A 6. sor végzi a kockák ellenőrzését. Ha a **szin** mezőben nulla van, akkor a sor nincs tele. Ezt ki kell bővítenünk azzal, hogy a sor csak akkor nincsen tele, ha a **szin** mezőben nulla van *és* ugyanakkor a **fal** mezőben is.

IF akna(x, y).szin = 0 AND akna(x, y).fal = 0 THEN…

Nagyszerű, most már be tudjuk tölteni a sorokat akkor is, ha fal van bennük; egy probléma van. Ha a sorok betöltése előtt kocka volt a fal fölött, akkor az berohan a fal mögé és eltűnik. Igen, mert a **zuditas** még a régi állapotról értesült; szóljunk neki. **5 IF akna(x, yy).fal = 0 THEN…**

Hiába, nincs változás. Próbáljuk ki **akna(x, yy - 1)**-gyel.

Ez se segít. Mi van, ha a **kockatorles**t is idehozzuk a következő sorból?

Se. Kénytelenek leszünk morfondírozni.

Először is állítsuk vissza a **zuditas**t korábbi állapotába, az a biztos. Töröljük az **IF** részt a **THEN**-ig.

Képzeljük el, hogy a vízszintes falra rátettünk egy-két elemet, s ezután betöltünk egy sort odalent. A **zuditas** mindent lejjebb visz, ami a betöltött sor fölött van, tehát a vízszintes falra tett kockáinkat is. Nomármost. Azt a kockát nem viheti lejjebb, amelyik a fal tégláján fekszik, mert akkor az belesüllyed a falba, és ezt ugye nem szabad. Ott kell hagynia. De a fölötte levő kockát se viheti lejjebb, mert akkor az meg eltünteti az alatta levő kockát. Tehát nem viheti lejjebb az efölött levő kockát sem, egyszóval: onnantól kezdve, hogy falra bukkantunk, egészen az akna tetejéig *egyetlen kockát sem* szabad lejjebb hozni. Persze csak abban az oszlopban, ahol a fal volt, mellette kétoldalt nyugodtan süllyedhetnek.

Ez lesz a megoldás, valósítsuk meg. A **zuditas** két egymásba ágyazott ciklussal megy végig az aknán, először egy sor mindegyik oszlopán, aztán a következő sor mindegyik oszlopán, és így tovább, alulról fölfelé. Vagyis amikor **yy** mondjuk 5, akkor neki még mindig tudnia kellene, hogy ebben és ebben az oszlopban falra talált, amikor **yy** még 10 volt. Tehát csinálnunk kell egy tömböt ennek számontartására. Tegyük be a rutinba:

3 DIM f(1 TO aknax)

Itt tartjuk majd a jelzést a falakról, ha valamelyik oszlopban falat találtunk, akkor 1-re állítjuk a tömb megfelelő elemét.

6 IF akna(x, yy).fal THEN f(x) = 1

Nullára visszaállítani soha nem kell, mert ez a tömb lokális, a **zuditas** minden egyes meghívásakor létrejön és a rutin lefutása után törlődik.

Ha **f(x)** valamelyik oszlopban 1, vagyis abban az oszlopban már volt fal, akkor az egész lejjebb hozatalt kihagyjuk. Tehát **IF f(x) = 0 THEN** és ide jön a tömbelem másolása és a **kockatorles**. De az első sorért felelős pótciklusba is bele kell tennünk ugyanezt.

Most jó! A falakra halmozott részek meg se moccannak, miközben minden egyéb lejjebb jön.

De még mindig nem tökéletes. A működésével nincs gond, csak a logikájával. Az akna alsó részét egy függőleges fallal kétfelé osztottuk. Nézzük meg az akna bal felét. Ugyanolyan fal határolja balról, mint jobbról? Ugyanolyan. Nézzük a jobb felét. Ugyanolyan fal határolja mindkét oldalról? Ugyanolyan. Nézzük meg az aknát följebb, ahol már nincs fal a belsejében. Ugyanolyan fal van mindkét ol-

dalán? Ugyanolyan. Akkor miért van az, hogy az akna bal vagy jobb felét nem elég betölteni ahhoz, hogy eltűnjön a tartalma, be kell hozzá tölteni a másik felét is? Úgy lenne logikus, hogy faltól falig kelljen betölteni, akkor tűnjön el.

Ez már komolyabb feladat. A **sorok** rutin jelen pillanatban végigmegy egy soron és mindenütt megnézi, hogy van-e ott üres kocka; ha nincs, meghívja a **zuditas**t, ami az illető sor fölött mindent lejjebb hoz, kivéve ahol fal van. Ha azt akarjuk, hogy csak faltól falig kelljen betölteni a sorokat, mindkét rutint át kell írnunk.

A **sorok**nak úgy kell működnie, hogy ha falat talál az aknában, akkor megnézi, hogy az addig megvizsgált részen volt-e üres kocka; ha nem, akkor átadja annak a sornak azt a részét a **zuditas**nak. Ezután továbbmegy a soron, megnézi a fal utáni részt is, ha megint falat talál, akkor átadja a **zuditas**nak, és így tovább. Először is beszéljük meg, hogy a **zuditas** kapni fog két további paramétert, amik megmondják, hogy a megadott sort mettől meddig zúdítsa le. Most nézzük a **sorok**at.

```
SUB sorok
FOR y = 1 TO aknay - 1
tele = 1
FOR x = 2 TO aknax - 1
IF akna(x, y).szin = 0 AND akna(x, y).fal = 0 THEN tele = 0: EXIT FOR
NEXT
IF tele THEN zuditas y
NEXT
END SUB
```

Tegyünk be egy vizsgálatot: ha fal van a vizsgált helyen, és **tele = 1**, akkor hívja meg a **zuditas**t az utolsóként talált faltól idáig. Mi az utolsóként talált fal? Amikor elkezdünk egy sort vizsgálni, akkor az akna bal fala, később pedig ahol falat találtunk, az. Tegyük el mondjuk **k** néven annak a kockának az **x** koordinátáját, ami az utolsóként talált faltól eggyel jobbra van; a 4. sorba, a **tele = 1** után beírhatjuk, hogy **k = 2**, hiszen a sor elején ez egész biztosan így lesz. Tegyük be az említett vizsgálatot 6. sornak:

```
IF akna(x, y).fal = 1 AND tele = 1 THEN zuditas y, k, x - 1: k = x + 1
```

K-t is itt állítjuk át, hiszen falat találtunk az aknában, a következő kocka tehát a legutóbbi fal utáni első kocka lesz.

Még át kell írnunk a 9. sort, mert ott csak **zuditas y** áll. Legyen helyette **zuditas y, 2, aknax - 1**, vagyis a teljes sor lezúdítása – ha nem volt fal a sorban, ez hajtódik végre.

Most írjuk át **zuditas**t is. Először is **SUB zuditas (y, k, v)** lesz a definíciója, ahol **k** a kezdő oszlop és **v** a végoszlop. Az **x** ciklus pedig ne 2-től **aknax - 1**-ig, hanem **k**-tól **v**-ig fusson, az 5. és a 12. sorban is.

Próbáljuk ki. A program majdnem jól működik: amikor a függőleges faltól balra eső részt betöltjük, eltűnik a fél sor. Éppen csak akkor nem tűnik el, amikor a *jobbra* eső részt töltjük be. Némi kísérletezéssel az is kiderül, hogy amikor az akna bal felét betöltjük, akkor a jobb fele is lejjebb jön, *ha* ugyanabban a sorban a jobb fél is be volt töltve, különben nem. Vagyis az akna jobb felében betöltött sor

eltüntetésére mindaddig várni kell, amíg a sor bal felét is be nem töltjük, de fordítva ez nem így van.

Az lehet a probléma, hogy ha a sor bal felében találunk egy helyet, ahol se kocka, se fal nincsen, akkor **tele = 0: EXIT FOR**, kilépünk, otthagyva csapot-papot. Ez megfelelt mindaddig, amíg a kőművesmunkát el nem kezdtük, de most már nem jó. Az **EXIT FOR**-t vegyük ki, és helyette következő sornak tegyük be, hogy **IF akna(x, y).fal = 1 THEN tele = 1**, vagyis minden falnál elölről kezdjük a sor betöltöttségének figyelését.

Most már majdnem jó, a sor jobb fele eltűnik önállóan, viszont viszi magával a bal felét. Persze, mert **k**-t nem állítottuk be ebben a legutóbbi sorban. De tegyük rendbe ezeket az **IF**-eket, jó? Kezdenek zavarossá válni.

```
IF akna(x, y).fal = 1 AND tele = 1 THEN zuditas y, k, x - 1: k = x + 1
IF akna(x, y).szin = 0 AND akna(x, y).fal = 0 THEN tele = 0
IF akna(x, y).fal = 1 THEN tele = 1
```

Ehelyett először is nézzük meg, hogy van-e fal. Ha van, akkor nézzük meg, hogy mit mutat **tele**, ha nincs, akkor nézzük meg a kockákat.

```
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
```

Ez máris áttekinthetőbb, és az **ELSE** után beállítottuk **k**-t is.

A sor bal fele azonban még mindig eltűnik, ha a jobb felét betöltjük. Debugoljunk. Tegyünk be **zuditas** mindkét hívása elé egy **STOP**-ot, aztán tegyünk pár kockát a bal részbe és töltsük be a jobbat.

Érdekes: a program már akkor megállt a 10. sorbeli **STOP**-nál, amikor a bal térfélen letettem az első elemet. Pedig sok zúdítanivaló ilyenkor még igazán nincs. Nézzük meg változóinkat. A megálláskor **y** 10 volt, **k** 18, **x** szintén 18. Ez elég furcsa. A 10. sor az a sor, ahol a vízszintes fal van, én pedig az akna fenekére tettem elemet. Lehetséges, hogy a **sorok** zúdítani akar egy vízszintes fal hatására? Nézzük csak. Ha falat talál és **tele = 0**, akkor 1-re állítja. Ha falat talál és **tele = 1**, akkor zúdít. Vagyis ha két faldarab közvetlenül egymás mellett van ugyanabban a sorban, akkor az egyik 1-re állítja a változót, a másik emiatt zúdítani akarja a semmit, ez így nem lesz jó, írjuk elő, hogy csak akkor zúdíthat, ha legalább egy kockát már megnézett a legutóbbi fal óta. Ennek a kockának az oszlopkoordinátáját **k** tartalmazza, ha tehát **x** nagyobb **k**-nál, akkor zúdíthat. **10 IF x > k THEN STOP: zuditas…**

Érdemes volt otthagyni a **STOP**-ot, mert az első elem lerakásakor még mindig megáll itt. Nézzük a változókat. Annyi a különbség, hogy **x** eggyel nagyobb, 19. Most nyilván ez történt: az első faldarab 1-re állította a **tele** változót, a második

észlelte, hogy **tele = 1**, de **x** nem volt nagyobb **k**-nál (egyenlők voltak), ezért nem történt semmi, a harmadik faldarabnál viszont már **x** nagyobb volt és zúdítani akart. Másképpen kell csinálnunk. Azt kell előírnunk, hogy az **x - 1**. kockában ne legyen fal. **10 IF akna(x - 1, y).fal = 0 THEN STOP…**

A program most már nem áll le ezen a helyen, amikor letesszük az első elemet; csak akkor áll meg, amikor betöltjük az akna jobb felét, de most már a 15. sorban. Nézzük, mi lesz. Természetesen maradt, ami volt: az akna bal fele is lejjebb megy. De most már tudunk debugolni. Nézzük meg a változókat. **Y** 19, **k** 19, **x** 32. Vagyis az akna jobb alsó sarkában vagyunk, **k** pedig a függőleges fal utáni oszlopra mutat. Ó, igen. A 15. sor **zuditas** utasítását is ugyanúgy kell paraméterezni, mint a 10. sorbelit: **y, k, x - 1**.

Most jó. Kivehetjük a **STOP**-okat és elégedetten hátradőlhetünk.

```
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, b%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racsinic (sz%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED rcs

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
```

```
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
```

```
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
rcs = 1
racsinic rcs

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
```

```
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
```

```
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, b
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
```

```
rcs = rcs + 1
racsinic rcs
END SUB

SUB racsinic (sz)
FOR x = 1 TO aknax
FOR y = 1 TO aknay
IF akna(x, y).tolt = 1 THEN akna(x, y).tolt = 0: akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

SELECT CASE sz
CASE 1: FOR y = 5 TO aknay - 1
FOR x = 14 TO 16
akna(x, y).tolt = 1
NEXT: NEXT
FOR y = 10 TO 19
FOR x = 2 TO 12
akna(x, y).tolt = 2
NEXT: NEXT
FOR y = aknay - 5 TO aknay - 1
akna(18, y).fal = 1
NEXT
FOR x = 17 TO 25
akna(x, 10).fal = 1
NEXT
CASE 2: FOR y = 5 TO aknay - 1
FOR x = 0 TO 2
akna(x + 2, y).tolt = 1
akna(x + 29, y).tolt = 1
NEXT: NEXT
END SELECT
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
```

```
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
END SUB
```

Pályázat

Most már háromféle fix elemünk van az aknában, ennyivel már egész komoly, érdekes pályákat alakíthatunk ki. Egyelőre csak az egy, aztán két zöld oszlopunk van, amik mellett folyamatosan jelen van egy piros téglalap és két faldarab. Apropó oszlopok, betöltötte már az Olvasó az első zöld oszlopot az utóbbi időben? Van

egy kis hiba ezen a téren, amikor az oszlop tartalmával együtt eltűnik, nem jelenik meg helyette a két oszlop, csak amikor lerakjuk az első kockát, illetve ha az elemet olyan helyen sétáltatjuk végig, ahol a két oszlop egyike van, akkor zöld kockákat húz maga után. Vagyis az aknában ott van az oszlop, csak nem rajzoltatjuk ki. Érdemes lesz egy **aknarajz**ot tenni a **racs** rutin végére.

Szóval pályákat. Mi lenne, ha ezek nem fixen beépítve lennének a programba, hanem egy külső file-ból venné őket, ahol a játékos át is tudja őket tervezni? Ez nagy előrelépés volna, lássuk, mi kell hozzá.

Egy szövegfile-ban könnyen meg tudjuk rajzolni a pályát, egyezményes jeleket használva. Tegyünk pontot annak a mezőnek a helyére, ahol nincs semmi különös, *-ot oda, ahol zöld rács van, / jelet oda, ahol piros rács, és # jelet oda, ahol fal. A külső falat hagyjuk ki az egészből, a játékos egyből kipróbálná, hogy lyukat üt rajta, mi meg kezelhetnénk ezeket az eseteket is, szóval minden sor legyen 30 mezőnyi széles, és legyen 19 sorunk. Így valahogy:

```
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
.......#....////....#.........
.......#....****....#.........
.......#....****....#.........
.......#....****....#.........
.......#....****....#.........
.......#....****....#.........
```

Ez egy olyan akna képe, aminek az alján egy téglalap alakú terület zöld, közvetlenül fölötte egy csík piros, és kétoldalt függőleges fal van a közelben. Másoljuk ezt ki egy szövegfile-ba, a *Jegyzettömb* tökéletesen meg fog felelni erre a célra, legyen a file neve mondjuk **tetris.lev**. (**Lev** mint *levels,* azaz szintek, pályák.) Még tegyünk az elejére egy jelzést, hogy ez az első pálya: legyen ez egy 1-es szám külön sorban.

Programunk innentől kezdve két file-ból áll, azazhogy még nem áll, mert a program még nem kezeli a file-t. Beszéljük rá. A **racsinic** végzi jelenleg a pályák közötti váltogatást, ennek a teljes szövege fölöslegessé vált; a **SELECT CASE** azért, mert ez készíti az elavult pályáinkat, az előtte levő ciklus pedig azért, mert fölösleges nullázni az **akna** tömböt, ha úgyis minden egyes elemét be fogjuk majd

állítani a pálya beolvasásakor. De fölösleges maga a **racsinic** is, mert a nevének már nincs köze a valódi eseményekhez, már nem rácsot inicializálunk, hanem pályát. Töröljük ki az egész rutint: nyomjunk *F2*-t, vigyük a csíkot a **racsinic**re és válasszuk a *Delete* parancsot.

Az **inic** és a **racs** rutin persze még meghívja **racsinic**et. Legyen az új rutin neve **palya**, és ne **rcs** legyen a paramétere, hanem **ply**. Sokat számít a program érthetősége szempontjából, ha a rutinok, változók nem viselnek elavult, korábbi tevékenységükre utaló nevet. Tehát cseréljük ki **rcs**-t **ply**-re, **racsinic** helyére pedig kerüljön **palya**. S most írjuk meg.

Először is meg kell nyitnunk a file-unkat:

OPEN "tetris.lev" FOR INPUT AS 2

Most keressük meg azt a pályarajzot, ami megfelel a kívánt pálya sorszámának. A file első sora egy számot tartalmaz, ami várakozásunk szerint megegyezik **ply** értékével. Olvassuk be és ellenőrizzük.

LINE INPUT #2, l$
IF VAL(l$) = ply THEN

Ha egyezik, akkor olvassunk be **aknay - 1** darab sort:

FOR y = 1 TO aknay - 1
LINE INPUT #2, l$

Menjünk végig rajtuk és vegyük ki sorban a karaktereiket:

FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)

Nullázzuk az **akna** tömb mindhárom mezőjét:

akna(x, y).szin = 0: akna(x, y).tolt = 0: akna(x, y).fal = 0

Nézzük meg, milyen karaktert kaptunk. Ha pontot, akkor nincs mit tenni, ezt ki is hagyhatjuk. Ha csillagot, akkor 1-est kell tennünk a **tolt** mezőbe:

SELECT CASE x$
CASE "*": akna(x, y).szin = 1

Ha viszont perjelet, akkor 2-est:

CASE "/": akna(x, y).szin = 2

Ha ellenben **#**-et, akkor a **fal** mezőbe kell 1-est tenni:

CASE "#": akna(x, y).fal = 1

Van még teendőnk? Nincsen, folytathatjuk a következő karakterrel, sorral, ha az utolsó sor is kész, akkor készen vagyunk.

END SELECT
NEXT: NEXT
CLOSE 2
EXIT SUB
END IF

Ha ellenben a beolvasott sorszám nem egyezik meg **ply** értékével, akkor folytatni kell a feldolgozást, vagyis tovább olvasni a file-t. Tehát kell egy **DO** a 4. sorba és egy **LOOP UNTIL EOF(2)** és egy **CLOSE 2** a rutin végére. Eléje még odaírhatjuk, hogy

```
FOR y = 1 TO aknay - 1
LINE INPUT #2, I$
NEXT
```

így az 5. sor következő végrehajtása megint egy számot fog olvasni, ha még van.

A programot elindítva azonnal megjelenik az új pályánk. Jól dolgoztunk.

De mégsem tökéletesen, mert amikor a zöld részt betöltjük, nem változik semmi. Valószínűleg nincs benne programhiba, csak hát ugye hiányzik a következő pálya képe. Rajzoljuk meg.

```
2
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
..............................
....................#####.....
......../////....////*****.....
```

A második pályát az a gondolat ihlette, hogy a játékos az első pálya teljesítéséhez bizonyára egy csomó kockát lerakott a zöld rács és a falak közé, és azt a területet most bepirosítottuk, hogy el kelljen tüntetnie a kockákat.

A dolog még nem az igazi: a zöld terület betöltése után képtelenségek jelennek meg a képernyőn. Az új pályát fölrajzolja, de eltűnnek a kockák a piros területről, viszont megmaradnak a zöldön. Nézzünk utána.

A **palya** rutin hajtódik ilyenkor végre. Először is neki nem szabad nulláznia a **szin** mezőt válogatás nélkül, hiszen ezzel minden kockát eltüntetne a képernyőről, ezt vegyük ki.

Már jobb, a piros részen ott maradnak a kockák, viszont a zöldről nem tűnnek el. Csakugyan, a boldogult **racsinic** végezte ezt a feladatot. Írjuk be a **racs** rutinba.

```
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT
```

Ez fog állni az első ciklusپár után. Nézzük meg újra.

Nagyjából már jó, csak a vízszintes piros csík ott marad. Tegyük be a **palya**ba, hogy a kockák nullázása előtt **kockatorles**t csináljon, ha fal vagy rács volt ott.

11 IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y

Munkánk most már jó – lehet pályákat tervezni!

```
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, b%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x
```

```
SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB
```

```
SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
```

```
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
```

```
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, b
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
END SELECT
NEXT: NEXT
```

```
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN akna(x, y).szin = 0: kockatorles x, y
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
aknarajz
sorok
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
```

```
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
END SUB
```

Távlati tervek

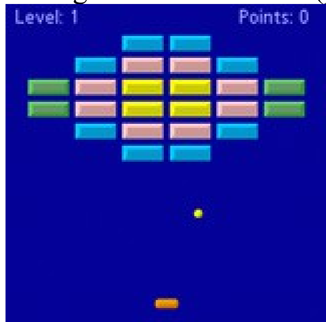
Tetrisünk azok közé a programozói feladatok közé tartozik, amikor nincs előre meghatározott, konkrét feladat, hanem szabadon szárnyalhat a képzeletünk. Ezek az igazán érdekes feladatok a programozó számára. Képzeletünket már eddig is engedtük szárnyalni valamennyire, de inkább csak kis nekirugaszkodásokat tettünk vele, kiszélesítettük az aknát, rácsot tettünk bele, a rácsot „színkeverőssé” tettük, végül falakat helyeztünk el. Közben azonban a programozó tovább agyal, egyre újabb és újabb megoldásokon töri a fejét. Szeretném most megismertetni az

Olvasóval azokat az ötleteket, amik eszembe jutottak, amíg az eddigi fejezeteken és a programon dolgoztam.

Jó lenne változatosabbá tenni a program grafikáját. Jelenleg minden elem egyforma kockákból áll, csak a színük különbözik; ezen jó lenne változtatni. Az elemszíneket úgy csináltuk meg, hogy minden színnek legyen egy sötétebb és egy világosabb változata, de ezeket még nem használtuk föl. Használjuk. Végső soron lehetne egy mintakészlet, ami annyi mintából állna, ahányat ki tudunk találni, és minden elem útnak indításakor a program választana egy mintát véletlenszerűen. A szín továbbra is egyértelműen következne az elem formájából, de a minta véletlenszerű volna. Amikor az elemeink megkövülnek az aknában, ott a minta is rögzül, így az aknában változatosabb kép alakul ki.

Később meg lehetne csinálni, hogy a mintákat ne a BASIC rajzolóutasításai csinálják, hanem a program kívülről vegye valamilyen file-ból. Ezáltal a játékos mintaszerkesztővé is válhatna, kicserélhetné a mintákat másfélékre, új mintákat hozhatna létre. Esetleg a paletta egy meghatározott részét fenntarthatnánk arra a célra, hogy a mintaszerkesztő-játékos tetszés szerint átdefiniálja, így saját mintái a saját színvilágához igazodjanak.

Korábban egy fejezet címéül leírtuk, hogy *Extrák,* de mindmáig csak egyes számban érvényes a szó, csak egyféle extránk van. Lehetne több is. A bombának lehetne olyan változata, ami nem egy kockát robbant ki, hanem függőlegesen az egész oszlopot, avagy csak addig, amíg lyukhoz nem ér. Vagyis ha egymás alatt van két üres hely ugyanabban az oszlopban, de köztük kockák vannak, akkor a felső lyukat meg tudjuk közelíteni egy ilyen bomba segítségével, de az alsóhoz még egy bomba kell. Lehetne olyan extránk, ami nem lyukaszt, hanem töm: betesszük egy körbeépített, de még megközelíthető üregbe, ott terjeszkedni kezd, mint valami hab, és valamekkora helyet kitölt. Persze némi véletlenszerűség lenne a dologban, így jó esély lesz rá, hogy egy-két fontos kocka üresen marad, immár esetleg megközelíthetetlenül. Ráadásul ha nincs olyan hely, ahol ez az extra jól kifejtheti hatását, akkor kénytelenek vagyunk olyan helyre ledobni, ahol a szétterjedő kockák esetleg csak kárt okoznak (például átterjednek a piros rács területére).



Speciális extra lehetne a *téglabda.* Ezt a szót sok évvel ezelőtt a *tégla* és a *labda* összevonásából alkottam az olyan játékok elnevezéséül, ahol egy ütővel ütögetett labda segítségével kell téglákat kiütögetni. Oldalt mutatok egy képet valamelyikükről; ezrével vannak ilyen programok. A mi esetünkben az ütő felül lehetne, s a labda az aknánk tartalmát ütögetné szét, amíg el nem veszítjük. Egy másik ősrégi játékprogramot is felhasználhatunk. Ebben felhőkarcolók fölött repül egy repülőgép vízszintesen, kor-



mányozni nem lehet, csak bombát ledobni, ami megsemmisíti a felhőkarcolókat. Itt a kockaoszlopokat semmisítené meg, esetleg csak bizonyos mélységig. A repülőgép egyre alacsonyabban repül, s ha nekimegy egy kockának, akkor elpusztul és folytatódik a játék tetrisként.

A legtöbb ötletem azonban az akna tartalmának kibővítésére van. Képzeljen el az Olvasó egy kis kapcsolót, ami valahol az aknában „lebeg", és ha leteszünk elé egy kockát, akkor a kapcsoló átkapcsol. Ha a kockát eltüntetjük onnan, megint átkapcsol, ha megint leteszünk, megint átkapcsol, és így tovább. Valahányszor a kapcsolót átkapcsoljuk, történik valami: például falak jelennek meg vagy tűnnek el, arrébb kerülnek a rácsok, bizonyos területekről eltűnnek a kockák, avagy éppen kockák jelennek meg ott, ahol addig nem voltak stb. Ugyanaz a kapcsoló mindig ugyanazt csinálná, tehát a játékos tudná, hogy a kapcsolónak mi a szerepe, és tudatosan törekedne arra, hogy építményét fölemelje odáig és bekapcsolja – avagy éppen ellenkezőleg, tudatosan próbálná elkerülni a kapcsolót, mert a hatása kellemetlen számára. Persze egyszerre több kapcsoló is lehetne az aknában, mindegyiknek meglenne a maga feladata. Némelyik kapcsoló esetleg csak más kapcsolók bekapcsolásának hatására jelenne meg vagy válna hozzáférhetővé (addig mondjuk fal mögött rejtőzne), így a pályák dupla logikai fejtörést igényelnének: először is ki kellene deríteni, hogy mire valók az egyes kapcsolók és milyen sorrendben célszerű bekapcsolni őket (hiszen ezt nem közölnénk sehol), azután pedig meg kellene oldani, hogy a kívánt sorrendben be is tudjuk őket kapcsolni úgy, hogy közben az építkezésünk se haladjon nemkívánatos irányba. Bizonyára gyakran előfordulna, hogy egy-egy fontos kapcsoló eléréséhez építeni kell egy külön emelvényt, amit aztán a kapcsoló átkapcsolása után le kell bontani, mert egyébként útban van.

Még nehezebb és izgalmasabb a játékos feladata, ha a kapcsolók nem akkor lépnek működésbe, amikor leteszünk oda egy elemet, hanem amikor egy bizonyos sort betöltünk. Például a jelenlegi fal rajzának elkészítenénk kétféle módosított változatát, az egyiken balra állna ki egy kis gomb, a másikon jobbra. (Lehetne egy harmadik is, ahol balra is meg jobbra is.) Ezt a faldarabot építenénk be helyenként a falba: a bal felé néző kapcsoló akkor lépne működésbe, amikor a tőle balra levő sort (sorrészt) betöltjük és az eltűnik.

A dolog úgy lenne a legszebb, ha a **tetris.lev** file-ban egy kis programnyelv segítségével lehetne megadni, hogy mi történjen az egyes kapcsolók megnyomásakor. Például az akna rajzában számokkal és betűkkel jeleznénk az egyes kapcsolók helyét, aztán a rajz után állna a magyarázat, hogy mit csinálnának az egyes kapcsolók. A számok és a betűk hivatkoznának rájuk, aztán pedig utasítások állnának egy BASIC-szerű nyelven leírva. Például egy részlet képzeletből:

```
.........
...###...
...#c#...
...###...
.........
c: WALL 3, 10 TO 8, 10
```

Itt a c-vel jelölt kapcsoló, ami valahol az aknában található, minden irányból falakkal van körülvéve, azokat tehát egy másik kapcsolónak kell eltüntetnie. A c kapcsoló működése lent olvasható: bekapcsolásakor fal húzódik a 3, 10-es koordinátától a 8, 10-esig. Lenne utasítás, ami zöld rácsot húz, lenne olyan, ami pirosat, lenne olyan, ami kockákkal tölt föl egy területet és olyan, ami üressel. Lenne feltételes utasítás, ami megvizsgálja, hogy egy adott koordinátahelyen mi van (fekete, piros, zöld, kocka feketén, kocka piroson, kocka zöldön vagy fal); hogy az illető kapcsolót hányadszor nyomjuk meg; hogy hány sort töltöttünk eddig be; és hasonlókat. Akár azt is megnézheti egy utasítás, hogy milyen elemet tettünk le legutóbb, így előfordulhat, hogy valamilyen működést csak bizonyos típusú elemekkel lehet előidézni. És persze a pontszámunkhoz is hozzájárulhat egyes kapcsolók megnyomása, lehetne utasítás, ami pontot ad vagy éppen elvesz.

Igaz is: a pontszám. Tetrisünk egyelőre egyáltalán nem számol pontokat, nem számolja a lerakott elemeket, a betöltött sorokat, semmit. Ezt majd azért csináljuk meg. Mondjuk járhatna pont minden egyes elem lerakásáért, nem sok, esetleg két pont. (Ha kizárólag páros számokat használunk, akkor csak páros szám lehet a pontszám. Így gyakrabban lesznek kerek pontszámok és általában megjegyezhetőbb a pontszám.) Az extrákért semmi. Minden sor betöltéséért járhatna mondjuk annyi pont, ahány kockából a sor áll, a teljes sor tehát harminc pont. Ha egy elemmel (mondjuk egy függőlegesen beillesztett **D**-vel, **H**-val vagy **I**-vel) egyszerre több sort tüntetünk el, akkor azért jutalmat kaphatnánk, mondjuk két sorért kétszer, háromért háromszor, négyért négyszer annyit, mint rendesen. Vagyis négy harminckockás sor eltüntetése külön-külön 120 pont, egyszerre viszont négyszer ennyi, azaz 480 pont. A zöld rács sikeres betöltéséért és eltüntetéséért kaphatnánk tíz pontot kockánként. A kapcsolók bekapcsolásáért semmit, ott az a jutalom (vagy a büntetés), hogy megtörténik, amit a kapcsoló csinál. És esetleg bizonyos pontszám elérésekor (mondjuk minden tízezredik pontnál) kapnánk valami speciális jutalmat. Egyes játékprogramok ilyenkor adnak egy életet, ezt mi nem tudjuk megcsinálni, de majd kitalálunk valamit. Mondjuk a következő három elemet a játékos tetszés szerint választhatná, avagy leradírozhatjuk az építmény legmagasabbra kinyúló részeit.

De ha számolunk pontokat, akkor azokat ki is kellene írni. A QB által a grafikus képernyőn használt betűtípus (az Olvasó látta már, amikor kiíratta egy-egy változó értékét a program leállásakor) túlságosan ormótlan, csúnya, készítenünk kell majd egy másikat. Ha azonban ezt megtesszük, akkor a számjegyek mellett egy

füst alatt tervezhetünk betűket is, és a program például az akna fenekén, a téglafalon kommunikálhat a játékossal. Minden pálya kezdetén megjelenhet egy felirat, a pálya neve például, és az egyes kapcsolók is adhatnak jótanácsokat a játékosnak.

Apropó feliratok: valamikor a munkálatok vége felé érdemes lesz egy szép címlapot szerkeszteni a programhoz, ez kiírná, hogy ő kicsoda (addigra jó lesz kitalálni neki egy nevet is, de ez a legnehezebb), és akár azt is megtehetné, hogy választást enged a **tetris.lev** és valamely más **.lev** file, azazhogy az összes hozzáférhető **.lev** file között. Így ugyanis a játékos megteheti, hogy szerkeszt egy sereg pályát, elnevezi **akarmi.lev**nek, és elküldi a barátainak, hogy próbálják ki ők is. Ez persze úgy szép, ha a rekordokat is nyilvántartjuk, mégpedig egy külön file-ban, ami a **.lev** mellé adható, a neve azonos vele (csak a kiterjesztés más), és ez feljegyzi, hogy milyen eredményeket értünk el: ki, mikor, melyik pályán, hány pontot.

De ha ezt megcsináljuk, akkor mást is csinálhatunk: filmet. Legyen egy file, amiben rögzítjük, hogy mi volt a pálya, aztán sorban az egyes elemekről, hogy mik voltak és merre vittük, hova tettük őket. Ezeket a file-okat aztán meg lehet nyitni a program főmenüjében (addigra már az is lesz neki), és lejátssza az egész játékunkat elejétől végig. Megtehetjük, hogy az elem összes ide-oda mozgatását mutassa, ha egy elemet ötször megsétáltattunk balra-jobbra az aknában, azt is, de azt is lehet, hogy ezeket csak magában számolgassa végig, az elemet aztán csak azon a helyen mutassa, ahova végül került.

Persze ne feledkezzünk meg az egyszerű, köznapi funkciókról sem, amilyen a következő elem mutatása; jelen pillanatban tetrisünknek fogalma sincs a következő elem mibenlétéről mindaddig, amíg a jelenlegit le nem tettük, de ennek nem muszáj így lennie, kisorsolhatjuk azt az elemet előbb is, és megmutathatjuk, hogy mi várható legközelebb. Sőt akár kisorsolhatunk egyszerre öt-tíz-akárhány elemet, és mutathatunk listát róluk valahol a sarokban, amikor egy elemet leteszünk, a lista első eleme megjelenik az aknában, az utolsó elem mögé pedig fölcsatlakozik egy akkor kisorsolt elem. A játékos így kicsit előbbre tervezhet.

Másik köznapi funkció az akna betelése. Mivel a mi aknánk széles, megtehetünk egy kis szívességet a játékosnak. Mielőtt az új elem megjelenik az aknában, ellenőrizzük, hogy nem ütközik-e ott nyomban, és ha igen, tegyük egy kicsit arrébb. Így a játék nem ér véget azonnal, amikor az építmény valahol eléri az akna tetejét, csak akkor, ha már sehol nincs hely letenni az új elemet.

Egyelőre ennyi jött ki az ötletrohamból – de még távolról sem merítettünk ki minden lehetőséget.

Pontozás

Kezdjük az előző fejezetben elmondottakat a pontozással. Ez az egyik legegyszerűbb feladat. Szedjük először is listába a pontot érő eseményeket.

Elem (nem extra) lerakása: 2 pont.

Sor betöltése: minden eltüntetett kockáért 1 pont.

Több sor betöltése: minden eltüntetett kockáért annyi pont, ahány sort egyszerre betöltöttünk.

Zöld rács eltüntetése: minden eltüntetett kockáért 10 pont.

Ennyi a teendő, lássuk, mihez fogjunk. Legjobb, ha a pontozást nem úgy csináljuk, hogy a program kérdéses helyein egy változót megnövelünk a szükséges számmal, hanem hívjunk inkább meg egy rutint, ami megcsinálja a növelést és elvégzi a további szükséges teendőket is: kiírja a megnövelt számot, ellenőrzi, hogy nem értünk-e el olyan pontszámot, amiért külön jutalom jár stb. Legyen a rutin neve **pontozas**, és legyen a jelenlegi pontszámhoz hozzáadandó összeg a paramétere, nevezzük **p**-nek. Ha valamiért le kell vonni a pontszámból, akkor negatív számot adunk paraméterül.

Legyen egy globális változónk **pont** néven, ez tárolja az aktuális pontszámot; írjuk be a főprogramba a **ply** után. Magának a **pontozas** rutinnak egyelőre nincs több dolga, mint **p**-t hozzáadni **pont**hoz, és – csúnya, nem csúnya, ez van – **PRINT** utasítással kiírni a képernyő bal felső sarkába.

```
pont = pont + p
LOCATE 1, 1: PRINT pont
```

Nézzük akkor, hova kell betenni a rutin hívásait. Két pontot adunk egy elem lerakásáért. Ez a **ragadas** rutin lesz; ennek 3. sora az extrát ellenőrzi, ezt követően bárhova betehetjük. A 17. sort választottam, ide kerül, hogy **pontozas 2**.

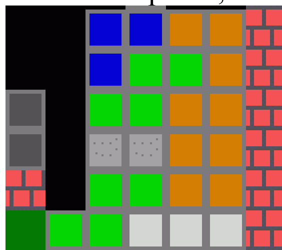
Működik, ámbár a pontszám elejét takarja a kőfal, betehetünk elé egy szóközt a **PRINT** utasításba. Nézzük a sor betöltését; ezt a **sorok** figyeli és a **zuditas** hajtja végre. Tegyük ez utóbbiba, mert ennek amúgy is tudnia kell, hogy hány kockát zúdít le. Csakugyan tudja: **k** és **v** jelzik a sor két végének koordinátáit, vesszük a kettő különbségét… nézzünk egy példát, ha **k** 1 és **v** 5, a különbség 4, de öt kockát töltöttünk be, tehát egyet hozzá kell adni. Illetve ha ez páratlan számra jönne ki, akkor adjunk hozzá egyet; ennek a legjobb módja, ha elosztjuk kettővel, hozzáadunk egy felet, aztán visszaszorozzuk kettővel. Az eredmény egész részét már nem kell vennünk, hiszen a **pontozas** úgyis egész paramétert vár. Rendben, betehetjük a **zuditas** elejére, hogy **pontozas ((v - k + 1) / 2 + .5) * 2**. Illetve ezt a QB így nem fogadja el, a rutinhívások kezelésének bizonyos misztikus okai folytán, ezért tegyük az egész kifejezést még egy zárójelbe.

A dolog nem jó; ha betöltjük az első pálya középső részének egy sorát, páratlan szám lesz az eredmény. Nyilván rosszul matekoztunk, de ha már kigondoltuk ezt a dolgot a páros számokkal, ragaszkodjunk hozzá. Vegyük ki ezt az osztogatást-szorozgatást, és a **pontozas** intézkedjen a páros számokról.

p = INT((p / 2) + .5) * 2

Így már jó. Csináljuk meg azt is, hogy ha több sor zúdult le egyszerre, szorzódjanak a pontok.

Ez azonban egy kicsit problémás. A sorok számát a **sorok** tudja megmondani, a hosszukat viszont a **zuditas**, és nekünk mindkét adatra szükségünk lenne; ráadásul van egy kis hiba, amit menet közben vettem észre. Próbálja ki az Olvasó, hogy a második pályán a vízszintes fal és az akna jobb fala közé eső részt úgy tölti be, hogy egyúttal a fölötte levő sornak is beteljen az a szakasza, ami a hézagba kerül. Mutatom képen is, mire gondolok.



Ha ebben a helyzetben egy **H** elemet állítunk a lyukba, akkor a falak közé kerülő rész eltűnik, helyére pedig öt kocka kerül: a **H** elem középső kockája, a két szürke és két narancssárga. Vagyis a sor teljesen tele van, mégse tűnik el. Ha ezután leteszünk egy elemet *bárhol az aknában,* akkor eltűnik, és helyére kerül öt kocka: a **H** elem felső kockája, két zöld és két narancs. Megint le kell tennünk egy elemet bárhol az aknában, hogy ez a sor is eltűnjön.

Persze a dolog végeredményben logikus. Amikor a **H** elemet letesszük, annak alsó kockája betölt egy sort – a középső és a felső azonban nem, mert ott a sor jóval hosszabb, és akkor még nincs betöltve. Amikor a sornak ez a része lesüllyed, falak közé kerül, és *akkor már* betöltött sornak számít, tehát el kellene tűnnie. De eddigre már a **sorok** régesrég végrehajtódott.

A dolgot úgy oldhatjuk meg, hogy a **zuditas** még egyszer hívja meg a **sorok**at. Ez elméletileg nem csinál végtelen ciklust, mert a **sorok** csak akkor hívja meg **zuditas**t, ha van eltüntetnivaló. Ha nem kell változtatni az akna tartalmán, akkor **sorok** szó nélkül kilép.

Így már jó, jöhet a többszörözés. Abban az esetben, amit föntebb bemutattam, a **sorok** és a **zuditas** többször is hívogathatja egymást, mire minden sor eltűnik, de azért úgy lenne helyénvaló, hogy ilyenkor is többszörözzön, elvégre egyetlen elem lerakása után tűnt el a több sor. A játékost az nem érdekli, hogy a mi rutinjaink hányszor hívogatták egymást. Ezért az lesz a célszerű, ha a rutinok kommunikálnak egymással pontozási ügyekben.

Morfondírozzunk. Amikor az elemet letesszük, lenullázhatunk egy globális változót, ami számon tartja, hogy hány sort zúdítottunk eddig le. Amikor lezúdítunk egy sort, eggyel megnöveljük a változót. A **zuditas** pedig ezzel a számmal beszorozza a kockák számáért járó pontot, így meg is vagyunk. Azaz lennénk, ha ez jó eredményt szolgáltatna, de nem fog, hiszen ha mondjuk négy sort zúdítunk le egyszerre, akkor az első sor kockáit még eggyel szorozza, a másodikéit kettővel, csak a negyedik sorra jön össze a négyes szorzó, amikor a **zuditas** már régesrég elszámolt az előző sorokkal. Ez így nem jó. Csináljuk azt, hogy a **zuditas** ne adjon egyetlen pontot se (vegyük ki belőle **pontozas** hívását), hanem csak szá-

molja, hogy hány kockát zúdít le, ezt tárolja egy globális változóban, ahogy azt is, hogy hány sorban voltak azok a kockák. A **ragadas** mindkét változót nullázza, mielőtt meghívja **sorok**at, és miután ez utóbbi elkészült feladatával, a változókban talált értékekből számolja ki a gyűjtött pontszámot.

Oké, ez így jónak látszik. Tehát legyen két új változónk, tegyük be őket a főprogramba: **zus** legyen a lezúdított sorok számlálója, **zuk** pedig a kockáké. **Ragadas**ban nullázzuk le őket a **sorok** meghívása előtt, utána pedig számoljunk el. Minden kockáért (**zuk**) annyi pont jár, ahány sort (**zus**) egyszerre betöltöttünk. Vagyis egészen egyszerűen **pontozas zus * zuk**ot kell írnunk.

Nagyszerű, épp csak egyetlen pontot se kapunk több betöltött sorért. Oppardon. Elfelejtettünk **zus**nak és **zuk**nak értéket adni. Ez legyen **zuditas** dolga: ha őt meghívjuk, akkor biztosan van lezúdítandó sor, tehát **zus = zus + 1** kerülhet az elejére, és a kockák számát megadhatjuk **zuk = zuk + v - k + 1**-gyel.

Hát bizony, így már jókora pontszámokat összeszedhetünk. A lelkiismeretes programozó utána is számol. Az első pálya jobb szélső része 9 kocka széles, tehát egy betöltött sorért 9, fölkerekítve 10 pontot kapunk; persze a pontszámunk 12-vel növekszik, hiszen az elem lerakásáért is jár pont. Két sor egyidejű betöltéséért 22-ről 60-ra ugrott a pontszámunk, ebből kettő az elemért, vagyis 60 – 24 = 36 pontot kaptunk. Két sort töltöttünk be, tehát soronként 18 pontot kaptunk – akkor stimmel, hiszen 9 kocka volt mindkét sorban. Három sor egyszerre: a pontszám 70-ről 154-re ugrott, kettő az elemért, tehát 154 – 72 = 82 pont. 82 osztva három sorral az 27 és egy kicsi, tehát igazából csak 81 pontot kaptunk, a többi a kerekítés, és akkor ez megint stimmel, hiszen 27 = 3·9. Négy sorért pedig 144 pontot kaptunk, osztva 4-gyel az 36, vagyis 4·9, tehát jól működik.

Csináljuk még meg a zöld rács eltüntetéséért járó jutalmat is. Ezért a **racs** rutin felelős; 11. sora éppen azt csinálja, hogy az akna minden kockájáról ellenőrzi, hogy ott a **tolt** mező 1-e, mert ha igen, akkor törölni kell a kockát belőle. Ide szépen betehetjük, hogy **pontozas 10**, és akkor ez kockánként végrehajtódik.

Az első pálya zöld területének betöltésével 12-ről (2 az elemért, tehát 14-ről) 214-re ugrott a pontszámunk, vagyis kereken 200 pontot kaptunk. Az elemek lerakásakor jól meg lehet számolni, hogy ez egy 4·5-ös téglalap, tehát 20 kockából áll. Helyes.

```
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, b%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
```

```
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
```

```
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply
```

```
paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT
'FOR p = 16 TO 84
'LINE (p * 2, 40)-STEP(1, 5), p, BF
'NEXT
'a$ = INPUT$(1)
'STOP
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
```

```
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
```

```
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, b
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
END SELECT
NEXT: NEXT
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
LOCATE 1, 1: PRINT " "; pont
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
```

```
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
pontozas 2
aknarajz
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
```

```
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Fontoskodás

Ha már megvan a pontszám, azt ki is kellene írni – ezt mondtam két fejezettel ezelőtt. Hát lássunk neki.

A QB által használt font csúnya, ormótlan, mindenképpen kerülném a használatát. Másik viszont nincsen, úgy értve, hogy a QB-nek nincs semmilyen eszköze a grafikus képernyőn használandó jelkészlet átállítására. Nekünk kell rajzolnunk.

Morfondírozzuk ki, hogyan tegyük. Legyen egy tömbünk, ami az egyes jelek képeit tárolja. A képeket szövegekként kódoljuk, például a QB **DRAW** utasításának megfelelő formában. Például **"d5r5"** lefelé, majd jobbra rajzol öt pontnyi hosszú vonalakat, így egy L betűt hoz létre. Ez jó lesz. Állapodjunk meg különböző számértékekben. A kockáink 10 pont magasak, tehát legfeljebb ekkorák lehet-

nek a betűk, ha azt akarjuk, hogy egy kockára ráférjenek. Legyenek inkább 9 pont magasak, akkor van egy kis elhatárolás a másik sortól – és a páratlan szám amúgy is jobb, mert így van egy középső pont, ahova például a 3-as számjegy középső vonalkája kerülhet. Ha 9 pont magasak, akkor legyenek 5 pont szélesek, ez remélhetőleg jól aránylik a kilenchez és szintén páratlan szám. Legyen a betűk közötti távolság egy pont. És írjunk mindezzel a legalsó sorba, az akna alsó téglafalára, ott nem fog zavarni.

A betűk ráérnek, egyelőre csak számjegyekre van szükségünk. Csináljunk olyasféle számjegyeket, mint a zsebszámológépek meg a digitális órák hétszegmenses kijelzői. Vagyis például a nulla úgy keletkezik, hogy a 9·5-ös téglalapot körberajzoljuk; a 8-as úgy, mint a nulla, de vízszintesen áthúzzuk egy vonallal; a 9-es és a 6-os a 8-asból keletkezik úgy, hogy a függőleges fal egy darabját kihagyjuk; stb. Mondjuk ki, hogy a betűk rajzolása előtt a grafikus kurzort a betű téglalapjának bal felső sarkába kell állítani; ez esetben a nullás számjegyet a **"d8r4u8l4"** kóddal tudjuk megrajzolni. Az egyeshez kell a **b** parancs is, ami rajzolás nélkül mozgatja a kurzort: **"br4d8"**. A kettes: **"r4d4l4d4r4"**. A hármas: **"r4d4nl4d4l4"** (itt kihasználtuk az **n** parancsot, ami a következő rajzolóparancs végrehajtása után visszatér). A négyes: **"d4r4nu4d4"**. Az ötös: **"nr4d4r4d4l4"**. A hatos: **"nr4d8r4u4l4"**. A hetes: **"r4d8"**. A nyolcas: **"r4d8l4u4nr4u4"**. A kilences pedig: **"nr4d4r4nu4d4l4"**.

Tegyük bele ezeket a szép szövegeket egypár **DATA** sorba, de úgy, hogy írjuk eléjük a számokat is, amiket rajzolni hivatottak. Ez sokat fog segíteni, ha már betűink is lesznek, de most se jön rosszul. Csináljunk nekik egy tömböt; legyen a neve **jel**, legyenek az elemeiben a **DRAW**-kódok, az indexeiben pedig a megfelelő karakterek ASCII-kódjai. Vagyis **jel(48)** a nullás számjegyet megrajzoló **"d8r4u8l4"** szöveget tartalmazza, mert 48 a nulla ASCII-kódja. Ennek megfelelően **DIM SHARED jel(0 TO 255) AS STRING** lesz a deklarációja.

Töltsük föl adatokkal. Irány az **inic**, ez olvassa be az adatokat. A kiaposztrófozott, most már fölösleges palettarajzolgatást kitöröltem, utána:

```
70 DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
```

A jelkészletünket tároló **DATA** sorok végére pedig kell egy **DATA "",""** végjel gyanánt.

Ezzel a jelkészletünket előkészítettük; csináljuk meg a **PRINT** utasítás „tetrises” megfelelőjét. A neve legyen mondjuk **iras**. Mik legyenek a paraméterei? Nyilván meg kell adnunk a szöveget. Meg kell adnunk a koordinátákat, ahonnan kezdve írni akarunk. Célszerű lesz a színt is megadni. Más paraméter valószínűleg nem fog kelleni. Legyen **t$** a szöveg, **x** és **y** a koordináták, **c** pedig a szín.

(A magyar programozó gyakran használ angol szavakra – jelen esetekben a *text* és a *color* szóra – utaló változóneveket, de rutin-, függvény-, mező- stb. neve-

ket is. Valamikor régen azt olvastam egy cikkben, hogy ez sznobizmus, de ezzel nem tudok egyetérteni. Szerintem egyszerű reflex. A BASIC kulcsszavai angolul vannak, a legtöbb amatőr programozó előbb-utóbb megtanul angolul, a profinak pedig muszáj. A szakkifejezések nagy része angolul van, vérbeli számítástechnikus soha nem mond olyat, hogy „átmásolom a parancsállományt a hajlékonylemezes meghajtóról a merevlemezre”, csakis azt, hogy „átmásolom a batchfile-t a floppyról a wincsire”, mert ő már akkor is így mondta, amikor a magyarra fordított szakkifejezések még nem léteztek, ezeket a fordításokat pedig nehézkesnek, erőltetettnek és nem utolsó sorban túl hosszúnak tartja. Abban az angollal átitatott légkörben, amiben a programozó tevékenykedik, természetes, hogy egy téglalaprajzoló rutinnak előbb adja a **box**, **frame** vagy **rec** nevet [utóbbi a *rectangle,* „téglalap” rövidítése, az első kettő dobozt, illetve keretet jelent], mint azt, hogy **teglalap**, mert ismer BASIC-nyelvjárásokat, amikben pontosan ugyanígy hívják az erre szolgáló utasításokat. Tetrisünkben a rutinok és függvények magyar neveket viselnek, a változók többsége szintén – de kizárólag azért, hogy az Olvasó könnyebben eligazodjon közöttük. Most azért választottam mégis angol neveket, hogy ezt a kis gondolatmenetet megoszthassam az Olvasóval.)

Az első feladat nyilván az, hogy végigmenjünk **t$** karakterein:

```
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
```

Emeljük ki a jelet megrajzoló szöveget a tömbünkből:

```
j$ = jel$(ASC(x$))
```

Mielőtt megrajzoljuk, állítsuk a grafikus kurzort a kívánt koordinátákra, persze rajzolás nélkül; állítsuk be a színt is; aztán rajzolhatunk.

```
DRAW "BM" + VARPTR$(x) + "," + VARPTR$(y) + "C" + VARPTR$(c) + j$
```

Ha ez megvan, akkor tovább kell vinnünk a kurzort. Abban állapodtunk meg, hogy a karakterek 5 pont szélesek, és egypontnyi hézag van közöttük. Tehát álljunk a következő karakter elejére:

```
x = x + 6
NEXT
```

Y-t nem kell változtatni, mert többsoros szövegek kiírását úgyse tervezzük.

Próbáljuk ki rutinunkat. A **pontozas** 5. sorában van a pontszám kiírása, ennek helyére kerüljön iras **STR$(pont), 0, 0, 15**.

Illegal function call hibaüzenetet kaptunk a **DRAW** utasításból. Lássuk, miért. Változóink értéke: **x** és **y** 0, **c** pedig 15, amennyit megadtunk. Viszont **j$** üres szöveget tartalmaz, nyilván azért, mert a **STR$** függvény mindig tesz egy szóközt a szám elé, és a szóközt nem definiáltuk a **jel$** tömbben. Ez azonban nem lehet probléma, hiszen a **DRAW** paramétere akkor sem üres szöveg, van benne egy **BM** és egy **C** utasítás.

Lehet, hogy ezek az utasítások nem szeretik a **VARPTR$**-t? Próbáljuk ki, írjuk be a számokat közvetlenül. **DRAW "BM0,0C15" + j$**.

Nini: egyből megjelent egy szám a képernyő bal felső sarkában. Akkor ez lesz a megoldás, szöveggé alakítani a számokat.

DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$

Most már megjelennek a számok, csak még nem tökéletesen, mert egymásra íródnak. Kiírás előtt töröljük őket: tegyünk a **pontozas**ba egy **aknarajz**ot az **iras** hívása elé.

Nem jó; az elemek lerakásakor továbbra is 2, 4, 8, 8, 10 a pontszámunk. Próbáljuk úgy, hogy áttesszük a feliratot a végleges helyére, a képernyő aljára. Melyik sarok legyen? Mondjuk a jobb alsó. Akkor **y**-t eszerint állítsuk be: a képernyő 200 képpont magas, a téglafal teteje 10 ponttal följebb van, vagyis 190, a számokat egy ponttal lejjebb tegyük, ez 191. **X** már nehezebb lesz: jobbra kell igazítani a számokat. 319 a legmagasabb koordináta, egy számjegy 6 pontot igényel, tehát be kell szoroznunk a számjegyek számát hattal; ez **319 - LEN(STR$(pont)) * 6** lesz.

Még az a gond, hogy időnként eltűnik a pontszám: a zöld rács betöltése után, extra ledobásakor. Tegyünk be egy-egy **pontozas 0** utasítást a visszarajzolásukhoz; kerüljön egy a **racs** rutin legvégére, egy pedig az **extra** 9. sorába.

```
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, b%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
```

```
DIM SHARED jel(0 TO 255) AS STRING

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
```

```
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
```

```
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
```

```
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
```

```
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, b)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, b
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
END SELECT
NEXT: NEXT
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT
```

```
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
pontozas 2
aknarajz
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
```

```
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
e = INT(RND * elemek + 1)
i = INT(RND * 4 + 1)
x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Sorakozó

Beszéltük, hogy mutatni kellene a következő elemet, sőt esetleg a következő néhány elemet, hogy a játékos tudja, mi vár rá és tudjon gazdálkodni. Ehhez először is az kell, hogy maga a program előre tudja, mik lesznek ezek az elemek. Csinálhatnánk tömböt, ami sorban tárolja az elemek adatait, de akkor valahányszor kivesszük a tömb első elemét, a többieket eggyel lejjebb kell hozni. Van erre gyorsabb módszer is: a szöveges változó.

Ha egy sereg adatot úgy akarunk sorba állítva tárolni, hogy egy pillanat alatt törölni lehessen közülük egyet, és ne kelljen a mögötte levőket egyenként arrébb másolgatni, hogy betöltsük a lyukat, akkor célszerű szöveges változóba tenni őket. Ha törölni akarjuk a legelső vagy a legutolsó adatot, csak levágjuk a szöveg megfelelő hosszúságú részét és kész. Ha egy közbensőt akarunk törölni, csak levágjuk a szöveg előtte és utána levő részeit, aztán a kettőt összeillesztjük.

Lássuk. A változónk legyen globális, a neve legyen **queue**; ezt a szót *kjui*nak kell ejteni és sorban állást jelent. Azért angolul, mert magyar nevet nem találtam neki; a *sor* nagyon sok mindenre utalhat, képernyősorokra, szövegsorokra, úgyhogy inkább ezt választottam. Dollárjel nem kell a végére, mert úgy deklaráljuk, hogy **DIM SHARED queue AS STRING**.

Most nézzük **ujelem**et. Négy változót állít be, de csak kettőt kell tárolnunk, **e**-t és **i**-t. Ha odaadjuk őket a **CHR$** függvénynek, máris szöveg lesz belőlük, és pontosan tudni fogjuk, hogy az első karakter **e**-t kódolja, a második **i**-t.

Ujelem két helyen hívódik meg **mozgas**ból: az elején, amikor a játék legelső elemét sorsoljuk ki, és a **ragadas** végrehajtása után, amikor új elemet választunk. Most azt kellene csinálnunk, hogy az első alkalommal azt mondjuk **ujelem**nek, hogy töltse fel **queue**-t valahány elem adataival, és később csak annyit mondunk neki, hogy vegye ki belőle az első elemet és sorsoljon egy újat **queue** végére. De van ennél egyszerűbb módszer is. **Ujelem**nek azt mondjuk, hogy ha **queue** a kívántnál kevesebb számú elemet tartalmaz, akkor töltse fel végig; ha nem, akkor csak adja oda az első elemét és tegyen a végére egy újat.

Mondjuk legyen **queue** hossza tíz elem, vagyis húsz karakter. Akkor ez kerül **ujelem** elejére:

```
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP
```

Feltöltöttük tíz elemig. Most vegyük ki az elsőt:

```
e = ASC(queue)
i = ASC(MID$(queue, 2))
```

és nyisszantsuk le az elejéről:

```
queue = MID$(queue, 3)
```

Végül tegyünk egy új elemet a végére – ez annyiból áll, hogy a fenti ciklusból idemásoljuk az értékadó sort. A korábbi négy értékadásból az **e**-nek és **i**-nek értéket adó sorok nem kellenek.

Egész biztos jól működik, csak nem látunk semmi változást – az elemeken nem látszik, hogy akkor sorsolták őket vagy előbb sorba kellett állniuk. A dolog értelme az volt, hogy így már mutatni tudjuk, hogy milyen elemek állnak sorban. Rajzoljuk ki őket.

Ha az alsó sorba, a téglafalra rajzoljuk az elemeket kicsinyítve, akkor nem zavarják a játékot és remélhetőleg jól látszanak. Bízzuk ezt is **ujelem**re. A végére írjuk a rajzolást. Amikor az eddigiek végrehajtódtak, akkor ott van **queue**, tartalmaz tíz darab elemet, ezeket ki kell rajzolni. Nézzük csak. Tíz képpont magas az a csík, amire rajzolhatunk. Négy képpont magas a legmagasabb elemünk (a függőlegesen álló **I**). Akkor rajzolhatunk minden elemet egy 2·2-es négyzetecske gyanánt, szépen ki fognak férni.

Először is menjünk végig az elemeken:

```
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
```

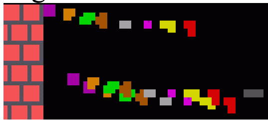
Ezzel kivettük az elemet. Most hozzuk ide **elemrajz** szövegét onnantól, hogy értéket adunk **ceruza$**-nak, de az első sorban cseréljük ki **e**-t és **i**-t **qe**-re és **qi**-re. A végére pedig tegyünk egy **NEXT**-et, hogy a **q** ciklus is le legyen zárva.

Dobjuk ki a két **kockarajz** utasítást, nem kockákat rajzolunk, hanem kicsinyített valamiket. A **"k"** sorban ez legyen helyette:

```
LINE (x, y)-STEP(1, 1), szin(qe), B
```

Mielőtt megcsináljuk az **"x"** sort is, próbáljuk ki. A bal felső sarok táján egy bizarr, összedobált valami jelenik meg. Csakugyan, hiszen a koordinátákat egyáltalán nem állítottuk be. Tegyünk a két **NEXT** közé egy **x = x + 5**-öt. Találomra választottam a számot, hátha elég lesz.

Valamik megjelennek, de azonnal kapunk egy **Subscript out of range**-et. Persze, mert **x**-et **ujelem** visszaadja mint az új elem egyik koordinátáját. Ezen könnyű segíteni: az **x**-nek és **y**-nak való értékadásokat hozzuk a rutin közepéről a legvégére.



Igen, most már csakugyan kapunk egy sorozat elemet, de mindenféle bizarr helyeken, és a sorozat kacskaringósan halad lefelé. A kép három elem lerakása után készült, három példányban mutatja a kirajzolt **queue**-t. Igen, a koordinátákat elemenként kell beállítani, mind a kettőt.

Kerüljön a **ceruza$**-nak való értékadás elé

```
x = q * 5
y = 195
```

Találomra választottam a számokat, meglátjuk, mire jutunk velük. 195 az aknafenék középvonala, erre próbáltam igazítani őket. Persze az **x = x + 5** most már nem kell.

Most már legalább tényleg sorakoznak, és nem íródnak egymásra, mert az **aknarajz** újrafesti a téglafalat és letörli őket. Csak szinte láthatatlanok szegénykék. Biztos, hogy 2·2-es négyzetekkel rajzoltunk?

Ó, dehogy rajzoltunk olyanokkal. Hiszen a ceruzakódok **x**-et és **y**-t egyesével állítgatják. Írjuk csak át, hogy kettőt adjanak hozzájuk és vonjanak le belőlük.

Már jobb, de jöjjenek egy kicsit följebb, mondjuk **y = 193**-ról induljunk.



Most jó. Leszámítva persze, hogy alig látszik szegényekből valami. Mi lenne, ha előzőleg törölnénk mögülük a téglafalat?

```
LINE (0, 191)-(100, 199), 0, BF
```



Nem rossz; talán nem az igazi, de egyelőre megteszi. Mondjuk a téglalap alsó szélét hozzuk föl 197-re, a bal oldalát pedig vigyük be 2-re, így mind a négy oldalán látszik az eredeti téglafal.

Még csináljuk meg, hogy az extra is látsszon. Az **"x"** sorba tegyünk egy **CIRCLE (x, y), 1, 8** utasítást, ez egy pirinyó, belül lyukas keresztet fog eredményezni pirosban.

Az viszont nem az igazi, hogy az éppen soros elem egészen a sarokban látszik. Esetleg fordítsuk meg a sorrendjüket, írjunk **x = 100 - q * 5**-öt. És hogy minden félreértést elkerüljünk, tegyünk a ciklus elé még egy **LINE (92, 191)-(100, 197), 1, BF** utasítást, ez a jobb szélső, vagyis legközelebb várható elemet sötétzölddel árnyékolja majd.

```
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
```

```
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF
SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": kockatorles x, y: IF akna(x, y).tolt THEN racsrajz x, y
END SELECT
NEXT
END SUB
```

```
SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
```

```
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
```

```
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB
```

```
SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, I$
IF VAL(I$) = ply THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, I$
FOR x = 2 TO aknax - 1
x$ = MID$(I$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
END SELECT
NEXT: NEXT
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, I$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
```

```
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
END SELECT
NEXT
pontozas 2
aknarajz
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
```

```
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Kapcsolásra készülj

Egy darabig kerülgettük azt a bizonyos kapcsolós dolgot… megcsináltuk a pontozást, a saját fontot, a következő tíz elem mutatását, de most már csak duráljuk neki magunkat és csináljuk meg. Lehet, hogy nehéz feladat lesz, de biztosan sikerül (eddig még minden tervünk sikerült), és nagyon megdobja majd a programot.

Kezdjük azzal a fajta kapcsolóval, ami az aknában lebeg valahol, és akkor kapcsol be, ha lerakunk elé egy kockát. (Az nem lenne jó, ha akkor kapcsolna be, ha *elviszünk előtte* egy kockát, mert akkor tetszés szerint kapcsolgathatnánk, nem kellene erőfeszítést tenni a bekapcsolásához.) A kapcsoló legyen az első pálya bal felén, a legalsó sorban, mert akkor a legelső elem lerakásával már bekapcsolhatjuk (hacsak nem extrával kezdődik a játék, akkor a másodikkal), nem kell külön azért építkezni, hogy bekapcsolhassuk. Tehát vegyük elő **tetris.lev**et és az első pálya legalsó sorába írjunk be baloldalt egy… mit is? Jelöljük a kapcsolókat számjegyekkel, akkor egy 1-est.

Első feladat: a **palya** rutin értesítése az eseményekről. Odáig stimmel a dolog, hogy beleteszünk egy **CASE "1" TO "9"** esetet az összes számjegy kezelésére, de hogyan tovább?

Tárolnunk kell, hogy egy adott kockán van-e kapcsoló. Ezt célszerű az **akna** tömbben tenni, de a **szin** mező nem alkalmas, mert akkor külön le kellene kezelnünk, hogy a kapcsoló nem kocka, nem akadály a kockáknak stb. A fallal is akkor boldogultunk, amikor áttettük külön mezőbe. A **tolt** mező se alkalmas, mert akkor az azzal foglalkozó programrészeket kellene értesíteni a változásról. A **fal** mező ugyanígy nem alkalmas, csináljunk csak szépen egy külön mezőt, legyen a neve **kapcs**, és a benne tárolt érték legyen a kapcsoló sorszáma. Vagyis ha a **tetris.lev** pályarajzában 1-est találtunk, akkor egy 1-es kerüljön a mezőbe. Tehát **CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)**.

Eddig jó. A következő alkalommal akkor találkozunk a kapcsolóval, amikor ki kell rajzolni. Ezt **aknarajz** végzi, tegyünk a **tolt** mezőt ellenőrző **SELECT CASE** mögé egy újabb vizsgálatot.

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs

Amint látható, új rutint bíztam meg a feladattal, nem **kockarajz**ot, éspedig azért, mert neki csak színkódokkal mondhattuk volna meg, hogy kapcsolót kell rajzolni, a különböző kapcsolóknak különböző színkódok kellettek volna, szóval macerás lett volna, „túlkódoltuk” volna a dolgot. Inkább legyen egy külön **kapcsrajz** a kapcsolók kirajzolására, paraméterei **x**, **y** és **k**, a kapcsoló sorszáma. Egyelőre bőven elég bele ennyi:

LINE (px, py)-STEP(9, 9), 8, B

Egy piros kocka, csak hogy tudjuk, hol a kapcsoló.

Most gondoskodjunk róla, hogy az elem mozgás közben ne törölje le. Az **elemtorles** gondoskodik az elem helyének visszaállításáról, miután azt elmozdítottuk, tegyük be ide is az **aknarajz**ban használt **IF**-et.

Ez is jó. Ha leteszünk egy elemet a kapcsolóra, a piros keret az elem fölött is látszik; ezt később majd megváltoztatjuk, mert úgy a logikus, hogy a kocka takarja a kapcsolót (az mintegy az akna „hátsó falára" van szerelve), de egyelőre megteszi.

Mostanra tehát megcsináltuk, hogy lett egy új képelemünk az aknában, ami nem befolyásolja a kockákat és azok sem őt. Most jön a neheze: kapcsolót csinálni belőle.

Azazhogy most még nem jön a neheze, mert ez még igazán egyszerű. A kapcsoló akkor kapcsol be, amikor leteszünk elé egy kockát. Ezt a **ragadas** figyeli, éspedig ott, ahol a **CASE "k"** eset van. Ide kell betenni: **IF akna(x, y).kapcs THEN STOP**. Merthogy egyelőre nem tudjuk, mi lesz a kapcsoló dolga és azt hogyan csinálja, tehát csak tegyünk be egy **STOP**-ot.

Működik: ha elébe rakunk egy elemet, megáll.

Természetesen a kapcsoló dolga az, hogy elindítson egy rutint, ami majd végrehajtja a kapcsolnivalókat. Legyen a neve stílszerűen **kapcsolo**, a paramétere pedig legyen a sorszám, tehát a **STOP** helyére **kapcsolo akna(x, y).kapcs** kerül.

Azazhogy álljunk meg egy pillanatra. Tegyük fel, hogy egy négykockás elemmel nyomjuk meg a gombot, éspedig úgy, hogy a kapcsolót történetesen az elem ceruzakód szerinti első kockája takarja le. Abban a pillanatban, ahogy az elem aknába kövesítésénél az első kockához érünk, a kapcsoló működésbe lép, és csinál valamit, ami esetleg függ a környező aknahelyek tartalmától is. A környék egy részén a most lerakott elem kockáinak kellene lennie, de nincsenek ott, mert a **ragadas** csak ezután tenné oda őket, ezért más lesz az eredmény. Máskor viszont a ceruzakód szerinti utolsó kocka keveredik a kapcsolóra, és akkor a kívánt eredményt kapjuk. Avagy a kapcsoló egy időigényesebb műveletet indít el, ami közben látszik, hogy még nincs minden kocka ott az aknában. Semmiképp se jó az a megoldás, hogy egy kocka lerakásakor azonnal elinduljon a kapcsolóhoz rendelt tevékenység.

Csináljuk inkább azt, hogy a **ragadas** jegyezze föl, hogy találkozott-e munkája során kapcsolóval (és melyikkel), és ha igen, az elem megkövesítése után nyomja a kapcsolót. Azazhogy a *kapcsolókat,* hiszen egy elem négy kockából is állhat, ha több kapcsolót tettünk le egymáshoz közel, akkor minden további nélkül bekapcsolhat egyszerre többet. Éppenséggel ez egyfajta nehezítés is lehet a játékos számára: egy fontos pozitívumot jelentő kapcsoló közvetlen közelében van egy másik, kellemetlen eredménnyel járó kapcsoló is, és nem mindegy, hogy milyen elemet használunk a jó kapcsoló megnyomásához, esetleg rálóg a rosszra is.

Tehát több kapcsolót kell megjegyezni. Erre remekül alkalmas egy szöveges változó, amibe sorban egymás után betesszük a **kapcs** mezőben talált adatokat: **IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)**.

Azt mondtuk, az elem megkövesítése után nyomogassuk meg a kapcsolókat. De mikor? A kövesítés a 17. sorbeli **NEXT**-tel véget ér. Utána jön a két pont hozzáadása, ezt még esetleg várjuk ki, aztán egy **aknarajz**, ezt is jobb végrehajtani,

utána viszont jön a **sorok** hívása. Előtte legyen a kapcsoló vagy utána? Ha előtte van, akkor a kapcsoló úgy fejti ki működését, ahogy az akna éppen van, és esetleg nem tűnnek el sorok azért, mert a kapcsoló például lyukat üt beléjük. Ez nem baj, éppenséggel alkalmas a játékos bosszantására is. Inkább azért célszerű, hogy a **sorok** előtt nyomjuk meg a kapcsolót, mert ha utána tesszük, akkor előfordulhat, hogy a kapcsoló megint szabaddá válik: letettünk egy elemet, ez megnyomja a kapcsolót, ugyanakkor eltüntet sorokat, azok eltűnnek, a kapcsoló ezáltal megint szabaddá válik, *aztán* kifejti működését. Ez hülyén nézne ki, tehát a **sorok** hívása előtt tegyük be az új rutinhívást: **kapcsolo be$**.

És most jön a morfondírozás. A kapcsoló természetesen nem csinálhat egy fix, programba épített, „bedrótozott” dolgot. A játékos adja majd meg az összes kapcsoló feladatát, pályánként külön-külön, egy speciális programnyelven. Tehát meg kell terveznünk a programnyelvet és meg kell írnunk hozzá a speciális értelmezőt.

Kezdjük a programnyelvvel.

A nyelvnek hirtelenjében a **Kate** nevet ötlöttem ki. Ez olyan, mint egy angol keresztnév, ejthető *két*nek is, de igazából azt jelenti: ***Ka****pcsolók a* ***Te****trisben.* Tehát tervezzük meg a Kate nyelvet.

Odáig már tudjuk, hogy lesznek egymástól elkülönített programrészek, amiket a kapcsoló jelével azonosítunk. A jelenlegi kapcsolónkat a pályarajzon egy 1-es szám jelöli, tehát lesz a programban egy 1-es címke, és utána, hogy minek kell történnie, ha a kapcsolót megnyomjuk. De jelölnünk kell, hogy hol van vége ennek a programrésznek. Jelölje a végét vagy a következő címke, vagy a következő pálya rajza. A címkéket azonosítsuk úgy, hogy egy karakter és utána egy kettőspont áll a sor elején, `1:` vagy `c:` stb., a következő pálya rajzát pedig egy sorban egyedül álló szám jelzi. Tehát a címkétől kezdve odáig dolgozzuk föl a programszöveget, amíg egy sor elején karakter plusz kettőspont vagy magányos szám áll. (Vagy vége van a file-nak.)

Első nekifutásra csináljunk néhány egyszerű utasítást, amik az eddig használt elemeket el tudják helyezni az aknában: kockát, rácsot, falat, üres helyet. Avagy legyen inkább csak egyetlen utasítás, aminek egyik paramétere az, hogy mit kell elhelyezni. Ez így egyszerűbb. Legyen az utasítás neve `PUT` (ejtsd: put), vagyis *tenni,* ez a kulcsszó számos BASIC-nyelvjárásban megtalálható. Legyen az első paramétere annak a kódja, amit le kell tenni:

0 – üres hely,
1 – zöld rács,
2 – piros rács,
3 – fal,
4 – kocka (mindegy, milyen színben, majd sorsolunk).

Legyen a második paramétere az **x**, a harmadik pedig az **y** koordináta. Egyelőre elég, ha az utasítás egyesével tud csak változtatni az aknahelyeken.

Ha a kulcsszó után, illetve a paraméterek között megkövetelünk egy bizonyos karaktert, akkor nagyon könnyű lesz a szöveget értelemszerűen szétvágni és feldolgozni. Ez a karakter nyilvánvalóan a szóköz lesz.

Annyit tehát már tudunk a Kate nyelvről, hogy ha például az 1-es kapcsoló megnyomásakor szeretnénk az akna 10, 15-ös pozíciójára kockát tenni, az így fog történni:

```
1: put 4 10 15
```

(A kis- és nagybetűket célszerű egyformának venni a kulcsszavakban, ez sok könnyebbséget jelent.)

Ennyi egyelőre elég ahhoz, hogy elkezdhessünk dolgozni az értelmezőn.

```
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING

inic
aknarajz
mozgas
```

```
DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
```

```
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
```

```
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
```

```
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
```

```
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB
```

```
SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
aknarajz
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
```

```
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
```

```
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Írjunk parsert

Ahhoz, hogy az írott szöveget megértsük, előbb el kell olvasnunk. A számítógép is így van ezzel. Minden értelmezőprogram tartalmaz egy *parser*t (ejtsd: parszer), vagyis a programszöveg elolvasására szolgáló programot – még a régi házigépes BASIC-értelmezők is, ahol a parser szót még nem használták, a programszöveg olvasását végző programrészek az értelmezőben szanaszét voltak. Később megjelentek a különálló parserek, amik először végigolvassák a programszöveget és átalakítják egy olyan belső formátumra, amivel az értelmező már jobban boldogul, mint az eredeti programszöveggel. (A szó egyébként a *parse* igéből ered, aminek jelentése: egy mondatot mondatrészekre, szófajokra bontani, nyelvtanilag elemezni; nyelvtanilag analizálni, értelmezni, kihüvelyezni az értelmet.)

A mi Kate nyelvünk parsere is legyen olyan, mint a házigépes BASIC-eké. Fölösleges erőpocsékolás lenne valamilyen köztes formátumra alakítani a kapcsolókhoz tartozó programszövegeket. A parserünknek nem lesz más dolga, mint a pálya beolvasásakor beolvasni és megfelelően tárolni az egyes kapcsolókhoz tartozó programszövegeket, majd pedig egy-egy kapcsoló megnyomásakor elolvasni és részekre szedni a megfelelő programszöveget, hogy végrehajthassuk.

Hogyan tároljuk a programszöveget? Azt is lehet, hogy egyáltalán nem tároljuk sehogyan, hanem amikor egy kapcsolót megnyom a játékos, akkor megint megnyitjuk a **tetris.lev**et és kikeressük benne a megfelelő pálya megfelelő kapcsolójához tartozó programszöveget; elméletileg ez nem jár érzékelhető idővesztéséggel, de azért előfordulhat. Például ha a program és a **tetris.lev** CD-n van, és a program indítása és a kapcsoló megnyomása között eltelt már néhány perc, akkor a CD motorja már valószínűleg áll, újra föl kell pörgetni, ez időbe telik, és a játékos meg is kérdezi magában, hogy ugyan mit pörgeti ez a CD-t éppen most. De igazából nem ezért akarnám tárolni a programszöveget, hanem egyszerűen azért, mert telik rá a memóriából, ezzel aztán igazán kár garasoskodni. Majd ha nem telik, akkor gondolkodunk más megoldáson.

Tehát tároljuk a programszöveget. Erre legjobb egy tömböt csinálni, legyen szöveges és legyen neki tíz eleme, 0-tól 9-ig számozva; a tömb indexe megfelel a kapcsolónak, a tartalma pedig a programszöveg. Feltöltését bízzuk a **palya** rutinra. Ez jelenleg megkeresi a soron következő pályát, beolvassa és dekódolja a tizenkilenc sornyi pályarajzot, aztán kilép. A 20. sorban van az a **CLOSE 2** utasítás, amivel lezárjuk a file-t; ez elé kell betennünk a programszöveget elolvasó részt.

Először is írjuk át a **tetris.lev**-et, hogy legyen mit olvasni:

```
.......#....****....#.........
...1...#....****....#.........
1: put 4 10 15
2
..............................
```

És olvassunk. Meddig olvassunk? Amíg egy sorban önálló számra nem bukkanunk vagy vége nincsen a file-nak.

DO UNTIL EOF(2)
LINE INPUT #2, I$

Honnan tudjuk, hogy mikor van önálló szám egy sorban? Ha a sor tartalmából a **VAL** függvény számot tud csinálni.

IF VAL(I$) THEN EXIT DO

Amíg nincsen ok a kilépésre, addig tegyünk el minden beolvasott sort a megfelelő tömbelembe. Igaz is, a tömböt még nem hoztuk létre. Tegyük be a főprogramba: **DIM SHARED prg(0 TO 9) AS STRING**. Tehát tegyük a beolvasott sorokat a megfelelő tömbelembe – de ehhez előbb tudni kell, hogy melyik tömbelembe valók. A **tetris.lev** programos részén kétféle sort engedünk meg: olyat, amelyik címkével kezdődik, és olyat, amelyik nem; ez utóbbi azért lehetséges, mert egy kapcsolóhoz több sornyi utasítást is rendelhetünk. Azt, hogy a sor elején címke van, onnan tudjuk meg, hogy a második karaktere kettőspont:

IF MID$(I$, 2, 1) = ":" THEN
i = VAL(I$)

I-be kerül az aktuális tömbindex. A sor többi részét pedig be kell tenni a tömbbe:

prg(i) = LTRIM$(MID$(I$, 3))

Ha a sor nem címkével kezdődik, akkor az egész megy be a tömbbe, de úgy, hogy hozzáírjuk a már ott levő szöveghez. Elválasztójelként használjuk a BASIC-ben szokásos utasításelválasztó kettőspontot; ha később majd megírjuk az utasításokat feldolgozó részt, ott megcsináljuk a kettőspont kezelését, és akkor a pályaszerkesztésnél nyugodtan írhatunk majd több utasítást is egy sorba, kettőspontokkal, a program nem érzékeli a különbséget aközött, hogy ő tette oda a kettőspontot vagy mi.

ELSE prg(i) = prg(i) + ":" + I$
END IF
LOOP

Eddig jó. Ki is próbálhatjuk: indítsuk el a programot, aztán állítsuk meg és nézzük meg **prg(1)** értékét. Persze előbb a **ragadas** 20. sorában levő **kapcsolo**-hívást ki kell iktatni, mert ott hibát jelez, nem ismeri ezt a rutint.

Sajnos nem egészen jó a dolog, a tömbelem tökéletesen üres. (Egy szövegváltozóról úgy lehet megtudni, hogy tényleg üres-e, vagy tartalmaz szóközöket, hogy előtte-utána kiíratunk bizonyos karaktereket. Én az *Immediate* ablakban szokásos rövidített szintaxissal **?"|"a$"|"** formában szoktam kiíratni egy-egy szöveges változót.)

Debugoljunk: tegyünk egy **STOP**-ot a **palya** rutinbeli **DO** elé. Hopp, meg is van a hiba. Azt mondtuk, hogy **IF VAL(I$) THEN EXIT DO**, ezzel akartuk elérni, hogy a program abbahagyja az olvasást, ha egy sorban önálló szám van; csakhogy a programsorunk is számmal kezdődik, az 1-es címkével, és a **VAL** abból is számot tud csinálni. Azért ez a sor így se rossz, csak tegyük át az utána következő **IF ELSE** ágába.

Máris jó: **prg(1)** értéke a `put 4 10 15` szöveg. Ez eddig rendben van; azért gyorsan teszteljük le, hogy jól kezeli-e a több kapcsolót és a többsoros programszövegeket is.

```
...1...#....****....#.........
1: put 4 10 15
2: put 3 12 15
put 3 13 15
put 3 14 15
2
..............................
```

Így írtam át a **tetris.lev**et; 2-es kapcsolót nem tettem a pályarajzba, úgyhogy az a programrész soha nem fog végrehajtódni, de ez nem baj, a lényeg, hogy a tömbelemekben a megfelelő értékek legyenek. A programot elindítva **prg(1)**-ben még mindig a korábbi értéket találjuk, **prg(2)**-ben pedig ezt: `put 3 12 15:put 3 13 15:put 3 14 15`. Még célszerű gyorsan megnézni **prg(0)**-t is, az ördög nem alszik, akármikor lenullázódhatott az az **i** és adat kerülhetett oda. Nem került, a tömbelem üres. A programszöveg beolvasása tehát hibátlan.

```
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
```

```
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
```

```
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
```

```
NEXT
FOR x = 1 TO aknax
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
```

```
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
```

```
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply THEN
FOR y = 1 TO aknay - 1
```

```
LINE INPUT #2, I$
FOR x = 2 TO aknax - 1
x$ = MID$(I$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, I$
IF MID$(I$, 2, 1) = ":" THEN
i = VAL(I$)
prg(i) = LTRIM$(MID$(I$, 3))
ELSE
IF VAL(I$) THEN EXIT DO
prg(i) = prg(i) + ":" + I$
END IF
LOOP
CLOSE 2
EXIT SUB
END IF
FOR y = 1 TO aknay - 1
LINE INPUT #2, I$
NEXT
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
```

```
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
aknarajz
'kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
```

```
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Értelmezz

Szépen előkészítettük a terepet a Kate nyelvű programszöveg számára: beolvastuk, megfelelően tároltuk, gondoskodtunk a végrehajtásáról. Már csak maga a végrehajtás van hátra.

Ez lesz a **ragadas**ban már meghívott **kapcsolo** rutin feladata. Csináljuk meg a rutint, akkor kivehetjük az **'**-ot a hívása elől. Egy paramétere van, **be$**, ami a benyomandó kapcsolók jeleit tartalmazza, abban a sorrendben, ahogy a ceruzakód szerint megrajzolt elem kockái alá kerültek. Később lehet szó arról, hogy végrehajtás előtt sorbarakjuk őket, de ez egyelőre ráér, úgyis csak egyetlen kapcsolónk van.

Maga **kapcsolo** nem kell, hogy sokat csináljon: végigmegy **be$**-on, beazonosítja a kapcsolókat, előveszi a szükséges programszöveget és végrehajtatja; de célszerű, ha magát a végrehajtást már egy másik rutin csinálja, mert később máshonnan is érkezhetnek programszövegeink, nemcsak a **prg** tömbből, illetve nemcsak egy **be$**-ban hivatkozott kapcsolólista alapján. Mindezek fényében így írtam meg a rutint:

```
DEFINT A-Z
SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = VAL(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB
```

Kapcsolo tehát csak kikeresi a szükséges programszöveget és átadja **program**nak, aminek már egy programszöveg a paramétere, mondjuk **p$**.

Ez a rutin fogja elvégezni a programszöveg részekre szedését. Amikor a jelenlegi körülmények között végrehajtódik, **p$**-ban a `put 4 10 15` szöveget kell találnia, ennek a feldolgozásával foglalkozzunk.

A szöveg részekre szedése úgy történik, hogy a szövegváltozóban tárolt folyamatos karaktersorozatban megkeressük a határoló szóközöket, és a köztük levő szövegrészeket különrakosgatjuk. Erre azonban nekem van egy rutinom, amit sok-sok éve használok, el is van téve egy kis file-ban külön:

```
DEFINT A-Z
FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION
```

A kis függvény a következőképpen működik. Argumentuma egy szöveges változó, aminek először is levág minden szóközt az elejéről, aztán megkeresi benne az első szóközt. Az **INSTR** függvénynek úgy adtam oda **s$**-t, hogy egy szóközt

hozzábiggyesztettem a végéhez, így a függvény akkor is talál szóközt, ha igazából nincs a változóban: a végén. Ezután a függvény visszaadott értékként kiveszi a szóköz előtti szövegrészt; ha a változó eredetileg nem tartalmazott szóközt, akkor az előző sorban hozzábiggyesztett szóköz jóvoltából az egész szöveg visszaadott érték lesz. De mielőtt vissza is adná ezt az értéket, az utolsó utasítás levágja **s$** elejéről ezt a részt, a szóközt is beleértve. Mivel a függvényeknek szabad módosítani az argumentumként kapott változók értékét, a függvényhívásból visszatérve a változóban az új szöveget találjuk. Mindebből tehát az lesz, hogy ha például a következő két utasítást hajtjuk végre:

```
abba$ = "Agnetha Benny Björn Anni"
elso$ = word$(abba$)
```

akkor **elso$**-ban az **"Agnetha"** szöveget találjuk majd, **abba$**-ban pedig csak a maradék **"Benny Björn Anni"** marad. Így **word$** ismételt hívogatásaival egy teljes szöveg feldolgozható. Persze arra vigyáznunk kell, hogy ha azért hívjuk **word$**-t, hogy beazonosítsa egy szöveg első szavát, mert azt valamiért meg akarjuk vizsgálni, de eldobni nem, akkor előzőleg tegyük át a szöveget egy másik változóba és azt adjuk oda neki. Persze azt is lehet, hogy **word$(abba$ + "")**-t írunk, akkor az argumentum már nem szöveges változó és nem lehet megváltoztatni az értékét.

Tehát a **program** rutin feladata az, hogy részekre szedje **p$**-t, és ebben **word$** fog segíteni neki. De mit csinál a részekkel? Valahova tennie kell őket. Csináljunk egy tömböt erre a célra, legyen szöveges, öt elemmel; ennél több szóból hátha nem fog állni egy utasítás. De magát az utasításszót ne is tegyük bele, az maradjon külön, csak a paraméterek sorakozzanak a tömbben. És legyen hozzá egy mutató is, ami megmondja, hogy hány paramétert tettünk a tömbbe.

```
DIM SHARED par(1 TO 5) AS STRING, mpar
```

Most már szétszedhetjük a szöveget.

```
DEFINT A-Z
SUB program (p$)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
```

Eddig megvagyunk. **Cmd$**-ba kerül a parancs kulcsszava (az angol *command,* „parancs” szóból), a **par** tömb 1-től **mpar**ig terjedő elemeibe pedig a paraméterek.

Most már jöhet a végrehajtás. Csináljunk erre is külön rutint? Miért ne?

```
exec cmd$
END SUB
```

Az **exec** (*execute,* „végrehajt”) rutin már sétagalopp. Nincs más dolga, mint ellenőrizni paraméterének tartalmát (előbb odaadva az **UCASE$** vagy **LCASE$**

függvények valamelyikének, nehogy gond legyen a kis- és nagybetűkből), és aszerint, hogy mi a szó, mást-mást kell csinálnia.

```
DEFINT A-Z
SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "PUT"
```

Mi itt a teendőnk? A paraméterek a **par** tömbben sorakoznak, közülük az első az elhelyezendő elem kódja. E szerint válogassunk. A 2. és a 3. tömbelem a koordináták.

```
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
END SELECT
END SELECT
```

Mivel **par** szöveges tömb, az adatokat **VAL**-lal kell kivenni belőle; ezért előzőleg tároltam a számmá alakított értékeket, hogy ne kelljen folyton a függvényt hívogatni.

Ha ez kész, akkor minden kész, munkánkat befejeztük. Próbáljuk ki.

Egész jól működik, csak nem csinál semmit, amikor megnyomjuk a kapcsolót. Tegyünk egy **STOP**-ot az **exec** elejére.

Megáll, de a **cmd$** teljesen üresnek bizonyul. Nézzük meg, miért. Tegyük át a **STOP**-ot a **program** elejére.

A **p$** is üres, márpedig **cmd$** értékét ebből kell kivágnunk. Nézzük meg, miért. Tegyük át a **STOP**-ot a **kapcsolo** elejére.

Itt **be$** egy kis arcocska gyanánt jelenik meg, ami az 1-es ASCII kódú karakter képe a DOS-ban. Hopp! Mi valamilyen okból a **VAL** függvénnyel alakítjuk számmá ezt a karaktert, holott az **ASC**-vel kellene. Javítsuk ki.

Ez az! Amint egy elemet leteszünk a kapcsolóra, piros tetriskocka jelenik meg a középső területen, a fal mellett. Kapcsolónk tehát működik.

De csak egyszer. Ha eltüntetjük a kapcsolót letakaró kockát és ismét leteszünk oda egy elemet (előbb persze eltüntetve a piros kockát is), már nem történik semmi. Persze, mert a **word$** úgy működik, ahogy mondtam: levagdossa a szöveg első szavait, amíg az el nem fogy. Javítsuk ki. Legyen a **program** paramétere **pr$**, és első utasítása az, hogy **p$ = pr$**.

Most már akárhányszor bekapcsolhatjuk a kapcsolót, újra meg újra.

Próbáljuk ki, működik-e a 2-es kapcsoló is. Tegyünk **tetris.lev**be egy 2-est is a pályarajzba.

Gond van: **Subscript out of range** üzenetet kapunk a **program** hetedik sorából. Nézzük, mennyi **mpar**: 6. Hogy szól **p$** maradék szövege? `"3 14 15"`. Mi van a **par** tömb elemeiben?

```
3
12
15:put
3
13
```

Vagy úgy. A kettőspontra nem készítettük föl. Először szét kell vágni utasításokra, aztán azokat egyenként feldolgozni. Csináljuk úgy, hogy a **program** még a teljes programszöveget kapja, de a **cmd$**-nak való értékadás előtt vágjuk ki az utasítást. A kettőspont előtti szövegrész levágása ugyanolyan dolog, mint a szóköz előtti szövegrész levágása, idehozhatjuk a **word$** függvény szövegét. Legyen most **s$** a teljes utasításlistát tartalmazó szöveg, **p$** pedig az éppen kivágott utasítás.

```
DEFINT A-Z
SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB
```

Így már működik a több utasítást tartalmazó kapcsoló is, a 2-es kapcsoló hatására faldarab rajzolódik az akna középső részére, egyik kockája lefedi a zöld rácsot. De azért a rács maradék részét be lehet tölteni, hiszen fal lerakásakor az **exec** nullázza a **tolt** mezőt, vagyis ott nincs is többé rács.

Ha viszont csakugyan betöltjük a rácsot, akkor kellemetlen meglepetésben lesz részünk: **Input past end of file** üzenet érkezik a **palya** rutin 35. sorából.

Ó, igen. Ha nem az első pályát akarjuk látni, akkor meg kell keresni a következő pálya elejét. Ehhez viszont át kell lépni mindent, ami előtte van. Két pályasorszámot eddig pontosan 19 sornyi rajz választott el, átugrottunk 19 sort és kész. Most már ez nem így van, a programszöveg is ott van közöttük, ami bármilyen hosszú lehet. Másképpen kell hozzálátni.

Töröljük ki ezt a ciklust, amiben a hiba keletkezett, és tegyünk a helyére valamit, ami addig olvas, amíg pályasorszámot nem talál. Probléma viszont, hogy utána a **LOOP** az 5. sorhoz küldi vissza a végrehajtást, ahol megint olvasunk és ellenőrzünk. Vagyis ha itt, a 6. sorbeli **IF** után beolvasunk valahány sort és az utolsóban pályaszámot találunk, akkor az 5. sor már az azutánit fogja olvasni, amiben persze nem lesz pályaszám. Másképp kell csinálni.

Mi van, ha a ciklus helyére nem teszünk semmit, hanem az 5. sorra bízzuk a beolvasást és a 6. sorra az ellenőrzést? Kicsit bizarr kép jön ki, a pálya felső részén egy csomó kapcsoló jelenik meg, a rácsok sokkal alacsonyabbak és a fal jóval lejjebb van. Igen, mert az első helyen, ahol kettessel kezdődő sort talált, úgy vette, hogy az a második pálya kezdete, aztán eszerint olvasott mindent. Talált egy csomó számjegyet is, hát kapcsolókat csinált belőlük. Tehet ő arról, hogy az a kettessel kezdődő sor az első pálya második kapcsolójának programja volt?…

De így már meg tudjuk oldani: tegyük be a 6. sorba, hogy **AND INSTR(I$, ":") = 0**, vagyis a sorban nincsen kettőspont.

A hibát kijavítottuk, a program megint jó.

```
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
```

```
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
DIM SHARED par(1 TO 5) AS STRING, mpar

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
```

```
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
END SELECT
END SELECT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).szin = 256
akna(aknax, y).szin = 256
NEXT
FOR x = 1 TO aknax
```

```
akna(x, aknay).szin = 256
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
```

```
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
```

```
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
```

```
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, l$
IF MID$(l$, 2, 1) = ":" THEN
i = VAL(l$)
prg(i) = LTRIM$(MID$(l$, 3))
ELSE
IF VAL(l$) THEN EXIT DO
prg(i) = prg(i) + ":" + l$
END IF
LOOP
CLOSE 2
EXIT SUB
END IF
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB
```

```
SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
aknarajz
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2
```

```
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1
END IF
END SELECT
NEXT
IF tele THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION
```

```
SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT
FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
sorok
END SUB
```

Tervezőasztal

A puding próbája az, hogy megeszik. Csak akkor tudjuk meg, hogy érdemes volt-e vacakolni a kapcsolókkal, ha tervezünk olyan pályákat, amik értelmesen hasznosítják őket, ahol érdekes játékot eredményeznek. Lássunk hozzá. Nyissuk meg a **tetris.lev**et egy szövegszerkesztőben és szépen dobjunk ki mindent, ami benne van, lejárt az ideje, nem kell már. Csináljunk új pályát.

Mondjuk legyen a közepén egy jó magas akna, teletöltve zöld ráccsal. Ezt persze így nem lehet betölteni, mert eltűnnek a sorok, de majd kapcsolóval megváltoztatjuk. Az akna két oldalán legyen egy-egy kapcsoló magasan a levegőben, olyan távolságra az akna falától, hogy egykönnyen ne lehessen elérni őket a fal tetejére illesztett elemekkel, egészen alulról kelljen építkezni. Az egyik kapcsolónak legyen az a feladata, hogy eltünteti az egyik falat, így a rács betölthető – a másik viszont tegye vissza a falat, hogy a játékos, aki lelkesen kipróbálja, hogy mi történik a másik kapcsoló hatására, utána mérgelődhessen és kapcsolhassa be a másik kapcsolót újra.

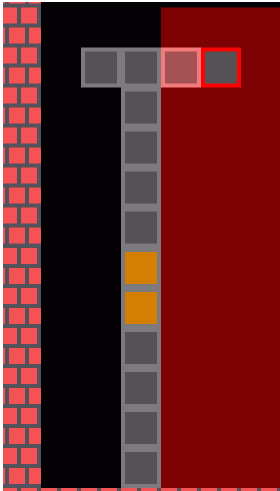
```
..............................
..............................
..............................
..............................
..............................
..............................
..........#********#..........
..........#********#..........
....1.....#********#.....2....
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
..........#********#..........
1: put 0 12 7:put 0 12 8:put 0 12 9:put 0 12 10:put 0 12
11
put 0 12 12:put 0 12 13:put 0 12 14:put 0 12 15:put 0 12
16
put 0 12 17:put 0 12 18:put 0 12 19
2: put 3 12 7:put 3 12 8:put 3 12 9:put 3 12 10:put 3 12
11
put 3 12 12:put 3 12 13:put 3 12 14:put 3 12 15:put 3 12
16
put 3 12 17:put 3 12 18:put 3 12 19
```

A dolog működik, kivéve, hogy a fal nem tűnik el, ott marad, csak az elemek törlik le útjukban. Szóljunk az **exec**nek, hogy adjon ki `PUT 0` utasításnál egy **kockatorles**t, és ha már itt vagyunk, akkor 1-es és 2-es rácsnál egy **racsrajz**ot, 3-asnál és 4-esnél pedig egy **kockarajz**ot, de ez utóbbiak elé is kell a **kockatorles**.

A pálya most már műszakilag működik, játékizgalom szempontjából nem egy durranás, de ez nem is csoda. Sok programozó van úgy vele: a programot megírja könnyedén, de pályát tervezni, figurákat rajzolni, urambocsá zenét szerezni hozzá már nem tud. Magam sem állok ezzel különbül. Hogy a játék mégis érdekesebb legyen, azt eszeltem ki, hogy a kapcsolókat piros ráccsal veszem körül, ami egy csíkban leér a földig, így a játékosnak mindenképpen le kell rombolnia az építményét, mielőtt a zöld rácsot betöltené.

Sajnos rögtön az első játékban sikerült becsapnom a saját pályámat: olyan építménnyel nyomtam meg a kapcsolót, amiből már csak egyetlen kockát kell kirobbantani, és máris szabad a rács. Magát a kapcsolót takaró kockát sem kell eltüntetni, mert pályarajzolási módszerünk fogyatékos: ugyanazon a helyen nem lehet egyszerre számjegy és törtvonal vagy csillag, így kapcsolót kizárólag fekete hát-

térre tudunk tenni, pirosra vagy zöldre nem. Alkalomadtán erről is gondoskodhatunk.



Rajzoltam egy második pályát is, egyelőre kapcsolók nélkül:

```
..............................
..............................
..............................
..............................
########/............/########
********/............/********
********/............/********
********/............/********
********/............/********
********/............/********
########/............/########
..............................
..............................
.............####.............
.............****.............
.............****.............
/////////////****/////////////
/////////////****/////////////
/////////////****/////////////
```

Itt három területet kellene betölteni, egymástól eléggé szeparáltan, s mindhármat piros rácsok határolják. Elég kevés olyan hely van a pályán, ahova le lehet tenni azokat az elemeket, amiket éppen nem tudunk értelmesen fölhasználni.

Viszont rögtön támadt két problémám. Először is a kapcsolók ott vannak a második pályán is, másodszor pedig amint egy elemet lerakok ezen a pályán, **Out of stack space** üzenetet kapok. Ez egy elég kellemetlen üzenet: betelt a BASIC veremtára.

Ilyenkor a programozó igencsak morcos lesz, mert az ilyen hibát nagyon nehéz felderíteni. A változók elvesztik értéküket, a BASIC elfelejti, hogy milyen rutinok hívogatták egymást, a *Calls* menü listája tehát üres lesz. Újra kell indítani a programot, és valahogy futás közben megtalálni azt a pillanatot, ahol elromlanak a dolgok.

A hiba akkor következett be, amikor letettünk egy elemet. De ha úgy akarjuk az okot földeríteni, hogy beteszünk egy **STOP**-ot a **ragadas**ba, akkor megőszülünk, mire betöltjük az első pályát, mert minden elem lerakásakor megáll. Persze megtehetjük, hogy átrajzoljuk az első pályát, hogy a zöld rács sokkal kisebb legyen. Esetleg próbáljuk meg ezt. Álljon a zöld rács csak egyetlen kockából, azt egy elemmel be lehet tölteni. A **tetris.lev**et le lehet másolni más néven, hogy később visszaállíthassuk az eredeti változatot.

A program most is ugyanazzal az üzenettel áll le a második pályára lerakott első kockánál. Most már betehetjük a **STOP**-ot a **ragadas**ba.

Odáig nincs gond, hogy végiglépkedünk a rutin 8–18. soraiban levő **FOR** cikluson. Jön a **pontozas**, ami meghívja **aknarajz**ot, itt a programozó türelmetlenné válik, a **STOP**-ot átteszi a **ragadas FOR** ciklusa mögé, újraindítja a programot és *F10*-zel átlépi **pontozas** hívását. Azért mindennek van határa. Itt azonban megint **aknarajz** következik; ezt esetleg töröljük ki, most hívtuk meg, fölösleges újra meghívni.

Következik a **kapcsolo** hívása, de ez a rutin nem csinál semmit, mert **be$**-ban üres szöveget talál. **Sorok** a következő rutin, ez a két egymásba ágyazott **FOR** ciklus miatt elég lassan lépeget, ezért a programozó beír a rutin legvégére egy **STOP**-ot, és megnyomja az *F5*-öt. Nicsak, nicsak… **Out of stack space.** Tehát **sorok** lesz a bűnös?

Ha **sorok** a bűnös, akkor arról lehet szó, hogy **sorok** és **zuditas** végtelen ciklusban hívogatják egymást. A veremtár valóban csak ilyesmitől telhet be. De hát arra nem gondolhattunk, hogy ez a két egymást hívogató rutin a bűnös, hiszen eddig is hívogatták egymást és nem volt gond. Mi változott?

Hát egypár dolog változott éppen. Lehet, hogy az első pálya feldolgozása során átállítunk valamit, amit később elfelejtünk visszaállítani, és emiatt keveredik a második pályán végtelen ciklusba. Lehet, hogy a második pálya rajzában van valami hiba. Próbáljuk ki például azt, hogy a **tetris.lev**ben fölcseréljük az 1-es és a 2-es számot, hogy elsőként a második pálya jelenjen meg.

A hibaüzenetet így is megkapjuk. Tehát csak a második pálya rajzában lehet a hiba. Még csak azzal sem függhet össze a probléma, hogy az első pálya kapcsolói a második pályán is megjelentek, hiszen most már nem jelennek meg.

Debugoljunk. Nézzük meg, hogy a **sorok y** ciklusának melyik pontján történik a baj. Tegyük be a **FOR y** után, hogy **PRINT y;**. Egy futtatás után kiderül, hogy ez így nem jó, mert *F4*-gyel képtelenség visszaszerezni az eredményeket, tehát **PRINT y; : a$ = INPUT$(1)** kell.

A program jó néhányszor elismételgeti az 1 2 3 4 5 számokat, aztán jön a hibaüzenet. Más szám nem jelenik meg.

Y ugyebár a sor száma, az 1–5. sorok a képernyő öt legfelső sora; a **tetris.lev**-ben utánanézve azt látjuk, hogy a legfelső vízszintes falak az 5. sorban vannak. Miért nem megy tovább az 5. sornál? Próbáljunk ki valamit: **IF y = 5 THEN STOP**. S ha megállt a program, lépegessünk *F8*-cal.

A **CASE 1** rész meghívja **zuditas**t, aztán az visszahívja **sorok**at, majd megint a **CASE 1** rész hívja meg **zuditas**t. Nézzük csak meg, milyen értékekkel! *Immediate* **?y k v**. Azt feleli: 5 2 1.

Tehát az 5. sorban kell zúdítani, a második oszloptól az elsőig. Ez így nem fog menni, ha **k** nagyobb **v**-nél, akkor a **zuditas**beli **FOR** ciklusok nem futnak le. De mit is akar zúdítani az 5. sorban, amikor ott semmiféle zúdítanivaló nincsen? Meg egyáltalán az egész aknában sehol? Próbáljuk ki azt, hogy a **zuditas** először is ellenőrizze, hogy **k** ne legyen nagyobb **v**-nél. **IF k > v THEN EXIT SUB**.

A program újraindítása és a **STOP** kiiktatása után kiderül, hogy nem oldottuk meg a problémát, az továbbra is jelentkezik. Akkor debugoljunk tovább: **zuditas** írja ki, hogy milyen paramétereket kapott.

Az 5 25 31 számok jelennek meg végtelen egymásutánban. Ez az ötödik sor jobb oldali része, éppen megfelel a vízszintes falnak. Ha az imént berakott **IF**-et kiiktatjuk, 5 2 1-et ír ki, szintén a végtelenségig ismétlődve; ez pedig a bal oldali vízszintes falnak felel meg. Tehát a vízszintes falak okozzák a problémát; de miért? Eddig nem tették.

Próbáljuk meg azt, hogy ha **zuditas** kizárólag falat talált az első megmozdítandó sorban, akkor **sorok**at ne hívja meg újból. Van benne egy **IF akna(x, yy).fal THEN f(x) = 1** sorunk, ennek a végére biggyesszük oda, hogy **ELSE csf = 0**, a két egymásba ágyazott ciklus elé pedig kerüljön **csf = 1**. Ez a változó azt fogja jelenteni: csak fal. És **sorok** hívása elé tegyük be, hogy **IF csf = 0 THEN**.

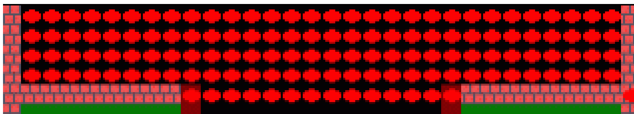
A dolog annyiban változott, hogy most az 5 2 1 és 5 25 31 számhármasok váltakozva jelennek meg, aztán betelik a verem. Alighanem **sorok**ban lesz a hiba, ő az, aki csökönyösen hívogatja **zuditas**t, nem képes túllépni az ötödik soron. Persze ha **zuditas**ban **'** mögé tesszük („kiremeljük”) **sorok** visszahívását, akkor minden megy, mint a parancsolat, csak ugye mi azért tettük oda azt a visszahívást, mert okunk volt rá.

Viszont mégis van valami haszna annak, ha **sorok** visszahívását kiremeljük. Feltűnően gyorsan záporoznak a pontok, a program meglepően adakozóvá válik. Egy-egy elem lerakása után a pontszámunk 58, 116, 174, 232 stb. pontra ugrik. Nyilvánvaló, hogy **zuditas** adja őket; hiába nem hívja vissza **sorok**at, ő maga mégis végrehajtódik, és legalábbis a pontszámok szempontjából azt hiszi, hogy zúdított. Minden elem lerakásakor 56 ponttal ajándékoz meg minket. Tehát tényleg nem az a baj, hogy **zuditas** meghívja **sorok**at, hanem az, hogy **sorok** fölöslegesen hívja meg **zuditas**t.

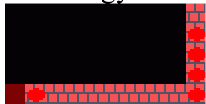
Lássuk, mikor teszi ezt. **Sorok**ban kétszer szerepel **zuditas** hívása, kezdjük az elsővel. 11. sor: **IF akna(x - 1, y).fal = 1 THEN** stb. Tegyünk csak ide a **THEN** mögé egy **STOP**-ot és nézzük meg, hogy hol áll le. A 2, 5 koordinájú pontnál. No de ha **x** 2, akkor **x - 1** pedig 1, az akna bal oldali fala, ott a **fal** mezőnek kell lennie. Nézzük csak… az akna külső falait az **inic** készíti el, hoppá! Itt még a **szin** mezőbe teszünk 256-okat. Írjuk csak át ezt gyorsan.

A hiba nem szűnt meg, továbbra is **Out of stack space**-t kapunk, anélkül hogy a 11. sor **STOP**-jánál megállnánk. Vagyis **zuditas** másik, 16. sorbeli hívásánál van a gond. Tegyünk ide is egy **STOP**-ot. A 32, 5 pozíciónál áll le. Ezt kicsit nehéz így bemérni, mi lenne, ha a **STOP** elé odaírnánk, hogy **kockarajz x, y, 255, 0**? Akkor a kérdéses helyen megjelenne egy bomba, pontosan jelezve a helyet.

A bomba a jobb oldali falon jelent meg, azon a ponton, ahol a felső vízszintes fal belefut a függőleges falba. Ezen a helyen **tele = 1**? Ugyan miért? Próbáljunk ki valamit. A **tele** változó kizárólag a **sorok**ban szerepel, az 5. és a 12. sor állítja az értékét 1-re, a 8. sor pedig 0-ra. Tegyünk be a 8. sorba egy olyan **kockarajz**ot, mint az előbb, akkor mindig, amikor **tele** nullára vált, megjelenik az illető helyen egy bomba.



Az akna felső öt sorának összes üres kockájában bombák jelentek meg. Oké, akkor tegyük át a **kockarajz**ot az 5. és a 12. sor végére: hol vált **tele** 1-re?



A sorok végén (az első sort kivéve) és a jobb oldali vízszintes fal első kockájában.

Mármost mit is jelent a **tele** változó? Azt, hogy a sor tele van kockával és le kell zúdítani. Melyik sor? Az **y**. sor **k**-tól **x - 1**-ig terjedő kockái. Miért vált 1-re a vízszintes fal bal oldali szélén, amikor előtte egyáltalán nincsenek is kockák? Ezt nyilván a 12. sor okozza, mert az 5. sor végrehajtásakor az akna bal szélén vagyunk. A 12. sor akkor hajtódik végre, ha először is **akna(x, y).fal = 1**, másodszor pedig **tele = 0**. Ezek a kritériumok itt csakugyan teljesülnek. Vegyük ki innen a **tele = 1** utasítást és nézzük meg, mi lesz.

Az lett, hogy a program vidáman működik, lehet vele játszani. De ellenőrizni kell, hogy a zúdítások rendben, szabályosan történnek-e. Vegyük ki a **STOP**-okat és töltögessünk be egypár sort.

Sajnos nem jól történnek a zúdítások. A középső vízszintes faltól jobbra eső szakaszt hiába töltjük be, nem tűnik el. Az a **tele = 1** tehát mégis kell oda. Tegyük vissza és küzdjünk tovább.

A programozó morcosan kidobálja a debugoláshoz szükséges utasításokat a **sorok**ból és gondolkodni kezd.

A **sorok** megpróbálja lezúdítani a vízszintes falat, amiben egyetlen szál kocka sincsen. Mi lenne, ha barátságosan megkérnénk, hogy ne tegye? Tartsuk számon,

hogy a kérdéses szakaszon volt-e kocka. Kerüljön a 4. sor végére egy **vk = 0** utasítás, hiszen a sor kezdetén még nem volt kockánk, a **CASE 0** sor végére pedig egy **: vk = 0 ELSE vk = 1**, hiszen ha a kocka színe nem 0, akkor ott kocka van. És **zuditas** mindkét hívása elé tegyük be, hogy **IF vk THEN**, vagyis csak akkor zúdítson, ha volt kocka a kérdéses szakaszon.

A program végre helyesen működik, de a programozó csak akkor hiszi ezt el neki, ha már mindent alaposan kipróbált. Az ilyen hibák a programozó rémei, nagyon alattomos, utálatos dolgok. A szerencse ezúttal kezünkre játszott: a programozó véletlenül rossz helyre tett egy elemet, ezért kénytelen volt betölteni a két alsó vízszintes fal közötti részt. Ahogy megtelt a sor, rögtön kiugrott a hibaüzenet.

A programozó gyanút fog. Valami nem stimmel, most más miatt kell jönnie. Az üzenet ugyanaz, **Out of stack space**, de nem jöhet azért, amiért eddig, mert nem az első elem lerakásakor jön, hanem amikor betöltünk egy sort. Amikor betöltünk egy sort? A programozó újraindítja a programot és kísérletezni kezd. Nicsak! Amint egy elemet letesz a vízszintes fal elé balra, jön az üzenet. Ez a piros rácson van, lehet, hogy megutálta a piros rácsot? Nem, mert ha az akna alján levő piros rácsra dobálunk elemeket, nem jön az üzenet. A vízszintes fallal van baja. Viszont az akna közepén, alul levő magányos vízszintes fallal nincsen baja, amellé lehet kockát rakni nyugodtan, sőt nincsen baja a bal oldali vízszintes falakkal sem, csak a két jobb oldali fallal, és azokkal is csak akkor, ha közvetlenül eléjük rakunk le kockát. Tehát ha olyan kocka kerül az aknába, amit folyamatos vízszintes fal köt össze az akna jobb szélével, azonnal jön a hibaüzenet.

A programozó újra átnézi a rutint, keresgéli, hol fordulhat elő, hogy **vk** állítgatása nincs összhangban az eseményekkel. Végül rábukkan a 11. sorra, itt átállítjuk **tele** és **k** értékét, de **vk**-ét nem. Legyen itt is nulla.

A program csodálatosképpen megjavul. A programozó megtörli homlokát és hátradől.

```
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
```

```
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
DIM SHARED par(1 TO 5) AS STRING, mpar

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
```

```
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END SELECT
END SUB
```

```
SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).fal = 1
akna(aknax, y).fal = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).fal = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
```

```
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
```

```
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN elemtorles e, i, x, y: x = x - 1
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN elemtorles e, i, x, y: x = x + 1
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
```

```
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
DO
LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, l$
IF MID$(l$, 2, 1) = ":" THEN
i = VAL(l$)
prg(i) = LTRIM$(MID$(l$, 3))
ELSE
IF VAL(l$) THEN EXIT DO
prg(i) = prg(i) + ":" + l$
END IF
LOOP
CLOSE 2
EXIT SUB
END IF
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
```

```
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
```

```
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2: vk = 0
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0: vk = 0 ELSE vk = 1
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN IF vk THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1: vk = 0
END IF
END SELECT
NEXT
IF tele THEN IF vk THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
```

```
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)

csf = 1
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1 ELSE csf = 0
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT

FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
IF csf = 0 THEN sorok
END SUB
```

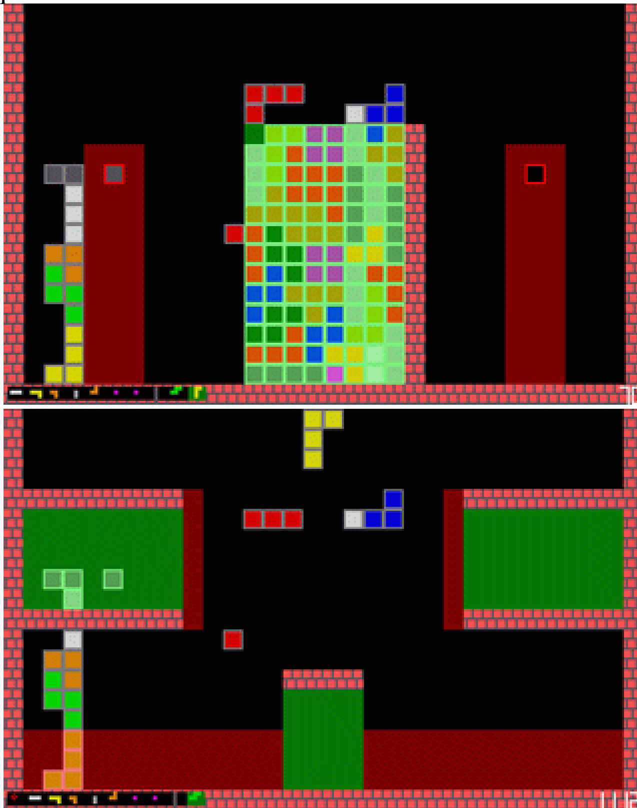
Pihenő

Most már visszatehetjük a **tetris.lev** korábban eltett változatát. Ez visszajuttat minket az előző problémához: az első pálya kapcsolói ott maradnak a második pályán is. Ez persze azért van, mert nem töröltük a szükséges tömbelemeket: a **kapcs** mezőt és a **prg**-t. Tegyük meg. A **palya** rutin az illetékes, tegyük be a 12. sorába a **kapcs** törlését, a legelejére (4. sornak) pedig ezt:

```
FOR a = 0 TO 9: prg(a) = "": NEXT
```

Ezt a problémát megoldottuk. Csináljunk most valami egyszerű kis dolgot, pihenjük ki az előző fejezet fáradalmait. Mondjuk tetrisezzünk.

Játszadozás közben könnyen kiderülhet, hogy a pályaváltások bajba keverhetik a játékost. Nézzük például a következő két képet: az első pálya befejezése előtt és után egy-egy pillanattal készültek.



A torony, amit az első pályán építettünk a kapcsoló eléréséhez, a második pályát teljesíthetetlenné teszi. Négy kockát takar a piros rácsból, de azokat sem bombával, sem a sorok betöltésével nem tudjuk eltüntetni. A bomba csak felülről lefelé használható, és nem tudunk a problémás kockák fölé bombát juttatni. A sorokat sem tölthetjük be, mert ehhez a bal alsó sarokba is elemeket kellene vinni. Azt sem tudunk. Semmit se tudunk csinálni, a pálya megnyerhetetlen.

Megoldást jelenthet, ha a bomba oldalra is működik. Akkor neki lehet vágni egy függőleges falnak, és abból robbant ki egy kockát. Próbáljuk ki, mi lesz ebben az esetben.

Amikor **mozgas** ellenőrzi, hogy az elem **mehet**-e oldalirányba, be kell tennünk egy **ELSE** ágat annak kezelésére, ha nem mehet. Ha nem mehet, akkor ol-

dalra robbantunk, mármint ha bombával van dolgunk. Ezt a ceruzakódból fogjuk tudni: ha **"x"** van benne, akkor bomba az illető.

Szedjük szét a 9. és a 11. sor **IF** utasításait két-két sorba, és mindkettő kapja meg bővítésnek ezt:

ELSE IF raktar(e, i) = "x" THEN akna(x - 1, y).szin = 0

illetve a második esetben **x + 1** áll.

Ez persze így még nem elég, hiszen a bomba ettől nem tűnik el, csak kiüti a kockát és továbbra is használható marad, ami azért túlzás. És ha a sor végére beteszünk egy **ujelem**-hívást?

Ez már segít, csak persze le kell törölni a kirobbantott kockát és a bomba helyét. Kell két hívás **kockatorles**re, persze **ujelem** hívása elé, az **x - 1, y** (**x + 1, y**) és az **x, y** koordinátákkal. Viszont a rács helyére ez fekete négyzetet tesz, tehát tegyünk melléjük **racsrajz**-hívásokat is.

A dolog működik. Ezzel nem tettük pehelykönnyűvé a fenti képeken mutatott helyzeteket, hacsak nem jön véletlenül óriási mennyiségű bomba egymás után. A játékos azt teheti, hogy lyukat üt a falon, azon átjuttat egy-két apró elemet, és azok segítségével betölti az alsó sort.

```
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE
```

```
CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
DIM SHARED par(1 TO 5) AS STRING, mpar

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
```

```
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END SELECT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB
```

```
SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).fal = 1
akna(aknax, y).fal = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).fal = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
```

```
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
```

```
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN
elemtorles e, i, x, y: x = x - 1
ELSE
IF raktar(e, i) = "x" THEN
akna(x - 1, y).szin = 0
kockatorles x - 1, y
kockatorles x, y
racsrajz x - 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN
elemtorles e, i, x, y: x = x + 1
ELSE
IF raktar(e, i) = "x" THEN
akna(x + 1, y).szin = 0
kockatorles x + 1, y
kockatorles x, y
racsrajz x + 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
```

```
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
FOR a = 0 TO 9: prg(a) = "": NEXT
DO
LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0: akna(x, y).kapcs = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, l$
IF MID$(l$, 2, 1) = ":" THEN
i = VAL(l$)
prg(i) = LTRIM$(MID$(l$, 3))
ELSE
IF VAL(l$) THEN EXIT DO
prg(i) = prg(i) + ":" + l$
END IF
LOOP
```

```
CLOSE 2
EXIT SUB
END IF
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
```

```
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2: vk = 0
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0: vk = 0 ELSE vk = 1
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN IF vk THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1: vk = 0
END IF
END SELECT
NEXT
IF tele THEN IF vk THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
```

```
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)

csf = 1
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1 ELSE csf = 0
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT

FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
IF csf = 0 THEN sorok
END SUB
```

Még több pihenő

Csináljunk egy új pályát, jó? Legyen benne négy kis fülke, és legyen négy kapcsoló egészen közel egymáshoz:

```
..............................
..............................
..............................
..............................
..............................
#######................#######
******#................#******
******#................#******
******#................#******
******#................#******
******#................#******
******#................#******
#######................#######
******#................#******
******#.......12.......#******
******#.......34.......#******
******#................#******
******#................#******
******#................#********
```

Mondjuk az 1-es kapcsoló nyisson kicsi ajtót a bal felső fülkén:

```
1: put 2 8 9: put 2 8 10
```

A 2-es nyisson ajtót a bal alsó fülkén:

```
2: put 2 8 16: put 2 8 17
```

Persze ez így még nem elég, a zöld területet két okból sem lehet betölteni: a zöld sorok eltűnnek, a középmagasságban nyíló ajtón át pedig nem tudjuk betölteni a kamra felső részét. De ez nem baj, majd teszünk róla. Pillanatnyilag fogalmunk sincs, hogyan, de teszünk.

A másik két kapcsoló nyisson ugyanígy ajtót a másik két fülkén:

```
3: put 2 25 16: put 2 25 17
4: put 2 25 9: put 2 25 10
```

Mielőtt folytatnánk, javítsunk ki egy hibát. Tesztelgetés közben a programozó véletlenül nekivezetett egy bombát a falnak oldalról, és az eltűnt. Persze az elemek ezután is fennakadtak rajta, mert a bomba nem nullázza a **fal** mezőt, de mivel meghívja **kockatorles**t, a kocka üresnek mutatkozik a következő **aknarajz**ig. Tegyünk ezért az **"x"** ceruzakód vizsgálata mellé egy **AND akna(x - 1, y).fal = 0** feltételt is (a második esetben persze **x + 1**-gyel).

Ez már jó, oldjuk meg, hogy a fülkéket teljesen be lehessen tölteni. Nem kell elkényeztetni a játékost, legyen csak nehéz dolga, mondjuk a fülkék belsejébe tegyünk egy-egy kapcsolót, amik átteszik az ajtókat a falak felső részére:

```
5: put 3 8 16: put 3 8 17: put 2 8 15: put 2 8 14
6: put 3 25 16: put 3 25 17: put 2 25 15: put 2 25 14
7: put 3 25 9: put 3 25 10: put 2 25 8: put 2 25 7
8: put 3 8 9: put 3 8 10: put 2 8 8: put 2 8 7
```

Jól összekutyultam, mindegyik fülkében olyan kapcsoló van, ami a másik fülke ajtaját viszi följebb. Oké, így a fülke egész területe elérhető, oldjuk meg, hogy be is lehessen tölteni őket. Mondjuk hozzuk kijjebb a függőleges falakat úgy, ahogy vannak: a bal oldalit egy kockával jobbra, a jobb oldalit eggyel balra. De ne `PUT` utasításokkal egyenként, egyrészt mert hosszadalmas, másrészt mert nem tudjuk, hogy éppen hol lesznek az ajtók. Csináljunk egy olyan utasítást, ami az akna egy darabját átmásolja máshová.

Legyen a neve `COPY`. Legyen az első két paramétere az átmásolandó terület bal felső sarkának *x* és *y* koordinátája, a másik két paramétere a jobb alsó sarok *x* és *y* koordinátája. És két további paraméter határozza meg, hogy a bal felső saroknak hova kell átmásolódnia. Tehát `COPY x1 y1 x2 y2 xn yn`, programozószokás szerint jelölgetve.

Az **exec** rutinba kell beírnunk a bővítést. Kerüljön bele egy **CASE "COPY"** ág, persze a `PUT` ága elé, az ábécérend miatt, és először is értékeljük ki a paramétereket, hogy könnyebb dolgunk legyen:

```
4 CASE "COPY": x1 = VAL(par(1)): y1 = VAL(par(2))
x2 = VAL(par(3)): y2 = VAL(par(4))
xn = VAL(par(5)): yn = VAL(par(6))
```

Ez egyből azt fogja jelenteni, hogy az 5-ig dimenzionált **par** tömbünk kicsi lesz, növesszük meg 10-ig.

No most. Le kell futtatnunk két egymásba ágyazott ciklust **x1**-től **x2**-ig és **y1**-től **y2**-ig, minden elemet átmásolva **xn, yn**-hez, de mielőtt ehhez hozzákezdünk, tegyünk egy kis óvintézkedést. A felhasználót semmi sem akadályozza abban, hogy az 1, 1-től 5, 5-ig húzódó területet `1 1 5 5` helyett `5 5 1 1`, sőt akár `1 5 5 1` formában adja meg, mert egy téglalapnak négy sarka van, bármely két átellenes sarok bármilyen sorrendben ugyanazt a téglalapot határozza meg. Tehát gondoskodjunk arról, hogy **x1** ne legyen nagyobb **x2**-nél, **y1** pedig **y2**-nél. Ez sok bosszúságot megtakarít.

```
IF x1 > x2 THEN SWAP x1, x2
IF y1 > y2 THEN SWAP y1, y2
```

Most már jöhetnek a ciklusok. Először persze **x1**-től **x2**-ig akarjuk vinni őket, aztán a programozó rájön, hogy így nehéz lesz kiszámolni az új koordinátát. Ha mondjuk 10-től 15-ig fut a ciklusunk, és 20-tól kezdve kell áthelyezni, akkor mennyi az új koordináta? De ha 0-tól fut a két érték különbségéig, azaz 5-ig, akkor az új koordináta egyszerűen 20 plusz a ciklusváltozó.

```
FOR x = 0 TO x2 - x1: FOR y = 0 TO y2 - y1
akna(xn + x, yn + y) = akna(x, y)
NEXT: NEXT
```

Itt fölhasználtuk, hogy egy rekord összes mezőjét egyetlen értékadással át lehet másolni egy azonos típusú rekordba.

Próbáljuk ki új utasításunkat. Legyen egy 9-es számú kapcsoló az akna alján (azért ott, hogy a teszteléshez ne kelljen építkeznünk), és csinálja ezt:

```
9: copy 8 6 8 19 9 6
```

Tehát a 8. oszlop 6. sorától a 8. oszlop 19. soráig (tehát egy függőleges oszlopból) másolunk át mindent a 9. oszlop 6. sorába, azaz egy kockával jobbra.

Egész jó, leszámítva, hogy **Subscript out of range** üzenetet kapunk a rekordokat átmásoló értékadásból. Nézzük, miért.

X és **y** egyaránt nulla. Persze, hát olyan nincs, hogy **akna(0, 0)**, szó se lehet róla. Ez helyesen **akna(x1 + x, y1 + y)** lesz.

Működik! Szépen jobbra viszi a falat, amitől az persze dupla vastag lesz, de majd töröljük mögüle a másikat. Ez már nem a `COPY` hatásköre, erre csináljunk külön utasítást. Feladat: egy téglalap betöltése valamilyen kockafajtával. Legyen a neve `FILL`, paraméterei pedig: elsőként a kockafajta (úgy, mint a `PUT` esetén), aztán a téglalap koordinátái, mint a `COPY` esetén.

Ez úgy lesz a legjobb, ha az egész `PUT`-ágat lemásoljuk az **exec**ben még egyszer, átírjuk `FILL`-re, és átírjuk a paraméterek kiértékelését. **X** és **y** helyett az első négy koordináta jön a `COPY`-ágból. Aztán hozzuk ide a két **IF**-et is a **SWAP** utasításokkal, végül a ciklusfejeket, és az ág végére a két **NEXT**-et. Jó ez így? Majd kiderül. Tegyük be a 9-es kapcsoló kódjába a `fill 0 8 6 8 19` utasítást is.

Hibaüzenet. Persze, a koordinátákat itt nem az első, hanem a második paramétertől kezdve adjuk meg, ki kell javítani a tömbindexeket.

Még mindig hibaüzenet, **Subscript out of range** onnan, hogy **akna(x, y).fal = 0**. Persze, a `COPY`-nál hozzáadtuk **x1**-et és **y1**-et, de itt nem. A **SELECT CASE VAL(par(1))** utasításban rengeteg hivatkozás van **x**-re és **y**-ra, ne ezeket írjuk át, hanem legyenek a ciklusváltozók **xx** és **yy**, és tegyük be, hogy **x = x1 + xx: y = y1 + yy**. Ez segíteni fog.

Segített is, már nem kapunk hibaüzenetet, és a csík szépen kifeketedik. Viszont a falakat is viszi magával, ami nem az igazi, a falaknak ott kéne maradni. Ezt persze könnyen megoldjuk: egy helyett két `FILL`-t használunk, mindkét fülkére egyet-egyet. De hátha van más megoldás.

Csinálhatunk egyfajta szűrőt is. A `FILL` betölti a kívánt téglalapot a kívánt kockafajtával, de csak akkor, ha bizonyos feltételek teljesülnek. Kétféle feltétel lehetséges: ha az adott mezőn egy bizonyos kockafajta van, illetve ha az adott mezőn *nem* egy bizonyos kockafajta van. De vagylagos feltételeket is el lehet képzelni, például csak olyan mezőket változtatunk meg, ahol piros vagy zöld rács van, avagy csak olyanokat, ahol vagy fal van, vagy kocka.

Leltározzuk a lehetséges feltételeket. Szűrhetünk arra, hogy egy mező üres, piros vagy zöld rácsot tartalmaz, kockát, kockát piros vagy zöld rácson, falat, avagy kapcsolót. Ezt jobb lesz nem számokkal megadni, mert akkor nehéz vagylagosítani őket, használjuk inkább a pályarajzon is alkalmazott jeleket. Például `.*` azokat

a mezőket cseréli majd, ahol üres mező vagy zöld rács van. A kockát is jelölnünk kell; legyen a jele üres mezőn `B`, zöld rácson `G`, piroson pedig `R` (a *block, green* és *red* szavakból, hiszen a Kate nyelv angol kulcsszavakat használ). Tehát `.*b` azokat a mezőket cseréli, amik vagy üresek, vagy betöltetlen zöld rácsot tartalmaznak, vagy olyan kockát, ami nem rácson van.

Oké, csináljuk ezt meg. Az **exec** 18. sorában kezdődő **SELECT CASE**-t legjobb lesz teljes egészében beágyazni egy **IF**-be, ami a feltételt értékeli ki. Magát a kiértékelést jobb lesz egy függvényre bízni.

```
IF filter(par(6), x, y) THEN
```

Filter szűrőt jelent, ez a függvény értékeli ki majd, hogy a megadott jelsorozatnak megfelelő kocka van-e az *x, y* koordinátákon. Hozzuk létre **f$, x, y** argumentumokkal, és nézzük, hogyan írhatjuk meg.

Nyilván végig kell menni **f$** minden karakterén:

```
FOR a = 1 TO LEN(f$)
x$ = UCASE$(MID$(f$, a, 1))
```

és ellenőrizni kell, hogy az aknahely tartalma megfelel-e a karakternek. Így oldottam meg:

```
SELECT CASE x$
CASE ".": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "*": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "/": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "#": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 1 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "B": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "G": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "R": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
END SELECT
NEXT
```

Illesszünk a **tetris.lev**beli `FILL` utasítás végére egy `.*bg/r` jelsorozatot és próbáljuk ki. (Ez tulajdonképpen azt jelenti, hogy bármi, ami nem fal vagy kapcsoló. Azon a részen kapcsolók úgysincsenek.)

No igen. Apró koncepciós hiba. A programozó dühösen mered a képernyőre. Szépen megcsináltunk egy komplett kis bővítést arra, hogy kiszűrjük a falakat, holott a letörölni való helyen *végig* fal van, az ajtókat kivéve. A világon semmi nem történik, nem is szabad semminek történnie.

Jól van, ha te így, én is úgy, mondja a programozó, és kitalál valamit. Mi lenne, ha a feltétel nemcsak a felülírandó aknahelyre vonatkozhatna, hanem egy másikra is? Mondjuk odaírhatnánk elé, hogy **x**, ez azt jelentené, hogy **x - 1**, vagyis a

bal oldali szomszéd. A jobb oldali szomszédot jelölhetné nagy **X**. Így már működne a kapcsolónk ezzel az utasítással.

Csináljuk meg. A **filter** függvény megkapja a koordinátákat, nincs más hátra, mint megváltoztatni azokat:

CASE "x": x = x - 1
CASE "X": x = x + 1

De akkor ki kell vennünk az **UCASE$**-t, ehelyett a többi betű mellé tegyük oda a kisbetűs megfelelőket. És vigyázat, az argumentumok megváltoztatása a hívó rutinban is változtat, tehát legyenek **filter** argumentumai **xx** és **yy**, az első sora pedig **x = xx: y = yy**.

A `FILL` végre működik, ámbár helyenként kissé zavaros az eredmény, ahogy az ember nyomogatja a kapcsolókat. Egy véletlenül a falnak vágott bomba még kuszább képet eredményez, úgyhogy jó lesz az **exec** végére egy **aknarajz**.

Ez sem oldja meg a problémát. Elmondom, mit csináltam. Megnyomtam a 9-es kapcsolót, ettől a fal eltolódott. Aztán megnyomtam az 1-2-3-4-es kapcsolókat, ettől megjelentek az ajtók, persze a fal *mögött,* az eredeti koordinátákon. Ez jó így, ne változtassuk meg, bosszankodjon csak a játékos. Azután megint megnyomtam a 9-es kapcsolót, és az ajtók is kijöttek az eltolt falra, hiszen a kapcsoló az eredeti fal oszlopából mindent jobbra visz, ami nem fal, tehát az ajtókat alkotó piros rácsot is. Ezután viszont az elemek letörlik a falat az ajtók alatt és fölött.

No persze. Ez logikus. Mit másol a `FILL` ebből az oszlopból? Mindent, ami nem fal, tehát az üres kockát is. Hova másolja az üres kockákat? Olyan mezőkre, ahol fal van. Mi jelenik meg egy olyan mezőn, ami korábban nem volt üres és most azzá tettük? Ugyanaz jelenik meg, ami korábban ott volt, mert a **kockarajz** nem töröl, csak rajzol. Az üres mező nem írja felül a többit.

Igen ám, de az **exec** hűségesen meghívja **kockatorles**t is! Hát akkor mi itt a baj?

Várjunk csak, nem, nem, dehogy. Az a `FILL` utasításnál van, hogy **kockatorles**t hívogatja, viszont a `COPY` végzi el ezt a másolást, és abba semmit sem tettünk a képernyő aktualizálására. A rutin végére rakott **aknarajz** erre szemlátomást nem elég, vegyük is ki, és a `COPY`-ágba tegyük be helyette 11. sornak, hogy **kockatorles xn + x, yn + y**.

Lám csak, a falunk máris eltűnik, ha a kapcsolókat 9, 1234, 9 sorrendben nyomogatjuk. Csakhogy mi nem ezt szerettük volna. Az volt a vágyunk, hogy amikor a 9-es kapcsolót másodszor megnyomjuk, az ajtó (a piros rács) jöjjön jobbra, felülírva a falat, az üres kockák viszont ne tegyék ezt. Szűrőt kell tennünk a `COPY`-ba is, hogy kívánságunk teljesüljön.

A `FILL` szűrőjét könnyű átvinni: a 10. és 11. sor egy **IF filter(par(7), x, y) THEN** alá vándorol. Azaz bocsánat, ebből 0, 0-k lesznek, tehát **xn + x, yn + x**. Azazhogy ebből meg az lesz, hogy letörlődik a fal teljesen, tehát **x1 + x, y1 + y**. Ez az, így már helyes!

A 9-es kapcsoló `COPY` utasítását kiegészíthetjük egy `*bg/r#`, vagyis „minden, ami nem üres kocka” jelentésű szöveggel, és készen vagyunk.

Csináljuk meg, hogy a jobb oldali fal is beljebb jöjjön. A 9-es kapcsolóhoz rendelhetjük ezt is, csak az oszlopkoordinátákat kell átírni:

```
copy 25 6 25 19 24 6 *bg/r#: fill 0 25 6 25 19 x.*bg/r
```

Ez nagyon szép, csak nem működik. Tényleg balra jön a fal, de ott marad mögötte is. Ja persze, az `x`. Nagy `X`-et kell írnunk helyette, mert most jobbra figyelünk.

```
DECLARE FUNCTION filter% (f$, x%, y%)
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
DIM SHARED par(1 TO 10) AS STRING, mpar
```

```
inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": kockarajz x, y, 255, 0
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
```

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "COPY": x1 = VAL(par(1)): y1 = VAL(par(2))
x2 = VAL(par(3)): y2 = VAL(par(4))
xn = VAL(par(5)): yn = VAL(par(6))
IF x1 > x2 THEN SWAP x1, x2
IF y1 > y2 THEN SWAP y1, y2
FOR x = 0 TO x2 - x1: FOR y = 0 TO y2 - y1
IF filter(par(7), x1 + x, y1 + y) THEN
akna(xn + x, yn + y) = akna(x1 + x, y1 + y)
kockatorles xn + x, yn + y
END IF
NEXT: NEXT
CASE "FILL": x1 = VAL(par(2)): y1 = VAL(par(3))
x2 = VAL(par(4)): y2 = VAL(par(5))
IF x1 > x2 THEN SWAP x1, x2
IF y1 > y2 THEN SWAP y1, y2
FOR xx = 0 TO x2 - x1: FOR yy = 0 TO y2 - y1
x = x1 + xx: y = y1 + yy
IF filter(par(6), x, y) THEN
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END IF
NEXT: NEXT
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
```

```
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END SELECT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
END SUB

FUNCTION filter (f$, xx, yy)
x = xx: y = yy
FOR a = 1 TO LEN(f$)
x$ = MID$(f$, a, 1)
SELECT CASE x$
CASE "x": x = x - 1
CASE "X": x = x + 1
CASE ".": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "*": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "/": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "#": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 1 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "B", "b": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "G", "g": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "R", "r": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).fal = 1
akna(aknax, y).fal = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).fal = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
```

```
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
```

```
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE 255
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
```

```
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN
elemtorles e, i, x, y: x = x - 1
ELSE
IF raktar(e, i) = "x" AND akna(x - 1, y).fal = 0 THEN
akna(x - 1, y).szin = 0
kockatorles x - 1, y
kockatorles x, y
racsrajz x - 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN
elemtorles e, i, x, y: x = x + 1
ELSE
IF raktar(e, i) = "x" AND akna(x + 1, y).fal = 0 THEN
akna(x + 1, y).szin = 0
kockatorles x + 1, y
kockatorles x, y
racsrajz x + 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END SELECT
CASE CHR$(0) + "P"
```

```
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
FOR a = 0 TO 9: prg(a) = "": NEXT
DO
LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0: akna(x, y).kapcs = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, l$
IF MID$(l$, 2, 1) = ":" THEN
i = VAL(l$)
prg(i) = LTRIM$(MID$(l$, 3))
ELSE
IF VAL(l$) THEN EXIT DO
prg(i) = prg(i) + ":" + l$
END IF
LOOP
CLOSE 2
EXIT SUB
END IF
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB
```

```
SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
```

```
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2: vk = 0
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0: vk = 0 ELSE vk = 1
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN IF vk THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1: vk = 0
END IF
END SELECT
NEXT
IF tele THEN IF vk THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
```

```
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)

csf = 1
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1 ELSE csf = 0
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT

FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
IF csf = 0 THEN sorok
END SUB
```

Fegyvertár

Miután visszaállítottam a pályák tervezés szerinti sorrendjét, újabb hibára bukkantam. Az első pályán a bal oldali kapcsoló eléréséhez emelvényt kell létrehozni, amiből a második pálya beköszöntével lesz egy fal. Ennek kilyukasztása miatt tanítottuk meg a bombát, hogy oldalra is működjön. De a falat csak kilyukasztani lehet, végleg eltüntetni nem, mert valahányszor betöltünk egy sort a képernyő alsó részén, a fal helyén újabb kocka jelenik meg, ugyanott és ugyanolyan színben, mint az előző. Végtelen utánpótlás van belőlük.

Logikus: a pályaváltáskor a vízszintes fal rárajzolódott a kockákra és nem tüntette el azokat. A kérdéses helyen tehát az **akna** tömb **szin** és **fal** mezője egyaránt

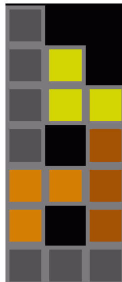
nullától különböző értéket tartalmaz. Sor betöltésekor a kockákat lejjebb hozzuk, akár van ugyanott fal, akár nincs; a program nem kalkulál azzal, hogy ugyanazon az aknahelyen kocka és fal is lehetne, mert eddig mi se kalkuláltunk vele.

Persze meg lehetne csinálni, hogy tényleg lehessen kocka és fal is ugyanott, miért ne, ez két egymástól független réteg lenne, de ez nem vezetne jó eredményre. Ha egymástól függetlenek, akkor a fal miért akadályozná a kockák mozgását, ha viszont nem akadályozza, akkor mire való? Éppen azért vezettük be a falakat, hogy akadályozzanak.

Jobb lesz tehát, ha továbbra is tiltjuk kocka és fal együttes előfordulását ugyanazon a helyen. A **palya** rutin fal betöltésekor törölje a kockát ugyanonnan. A 17. sorba kell egy **akna(x, y).szin = 0** utasítás.

Most azonban kérdeznék valamit. Megpróbálta az Olvasó a játék jelenlegi pályáit megnyerni így, ahogy vannak, egymás után? Őszintén bevallom, hogy a második pályán nem jutok túl. Maga a pálya nem nehéz, de az első pályáról ott marad az alján egy csomó szemét, amiket el kell tüntetni, hogy az alsó piros tartomány fölszabaduljon. Miközben ezzel küzdök, rengeteg alkalmatlan elem érkezik, amiket valahol fel kell halmoznom, így egyrészt állandóan kockahalom áll a pálya alsó részén, másrészt kockákkal telik meg az összes vízszintes fal teteje. A középső falra halmozott kockák akadályozzák a mozgást a pályán, a két felső falra rakottak pedig csak Isten a megmondhatója, mikor és hogyan tűnnek majd el onnan.

Ha nem akarjuk, hogy a játékos a második pályán a falhoz vágja a tetrisünket, adnunk kell a kezébe valamit, amivel fölléphet a mindent elborító kockahalmok ellen. Mondjuk egy-két extrát.



Próbáljuk ki például azt a fajta bombát, ami a legelső lyukig üti át a kockafalat. Ezzel például egy ilyen szituációban két bomba segítségével hozzáférhetővé tehető az alsó lyuk: az első kirobbantja a felső lyuk fölötti két sárga kockát, a második az alatta levő narancssárgát. Ha viszont a két szélső oszlop valamelyikére dobjuk le, akkor az egész oszlop egy bombától eltűnik, és oldalról megközelíthetjük a lyukakat. Nagyon hatékony eszköz lehet.

Mielőtt magát a bombát megcsinálnánk, beszéljünk meg valamit. A **queue** jelenleg azt tárolja, hogy egy sorban álló elem extra lesz-e. Ha már többféle extránk van, akkor el kell döntenünk, hogy az extra fajtáját mikor válasszuk ki: amikor a **queue**-ba kerül vagy amikor megjelenik a képernyőn. Ha csak akkor választunk fajtát, amikor az extra megjelenik a képernyőn, akkor nem kell a **queue**-ban az extra fajtáját is tárolni, sőt azon sem kell törnünk a fejünket, hogyan ábrázoljuk alul kicsinyítve a különféle extrákat. Viszont amikor az extra megjelenik a képernyőn, akkor már mindenképpen választanunk kell neki egy fajtát, mert nem lenne illendő, hogy mindegyik egyformán jelenjen meg az aknában. Akkor a játékos csak akkor tudná meg, milyen fegyver van a kezében, amikor már elsütötte.

Tehát amikor az extra megjelenik az aknában, válasszunk neki fajtát. Ez célszerűen egy globális változó segítségével történhet, legyen a neve **ext**, én a **ply**-t deklaráló sorba írtam be a nevét. Értéket mondjuk az **ujelem**ben kaphat, itt világos a szituáció: új elemet sorsolunk ki. A legegyszerűbb, ha a rutin legvégén véletlenszerűen értéket adunk **ext**-nek. Még azt se kell vizsgálnunk, hogy a soron levő elemünk extra-e – ha nem az, úgyse csinálunk semmit ezzel a változóval.

ext = INT(RND * dbext) + 1

A rutin végére írtam ezt az utasítást, ami pedig **dbext**-et illeti, ő egy konstans lesz, az értéke jelenleg 2, mert kétféle extránk van. Azaz lesz.

Folytassuk az extra kirajzolásával. Ezt jelenleg **kockarajz** végzi, akkor, ha a **szin** paraméterben 255-öt kap. Ha többféle extránk van, akkor macerás mindegyikhez egy színkódot rendelni, csináljuk másképpen. A 255-ös színkódot **elemrajz**tól kapja **kockarajz**, itt egyszerűen kicserélhetjük az egész utasítást **extrarajz x, y**-ra, innentől külön rutin végzi majd az extrák rajzolását.

Az új rutin először is átveszi **kockarajz** első sorát, hiszen **px** és **py** itt is kell, aztán elágazik egy **SELECT CASE ext** utasítással, és a **CASE 1** ágba máris áthozhatjuk a **kockarajz** 255-ös ágának szövegét, onnan pedig kitörölhetjük az ágat. **CASE 2**-nek pedig használjuk ugyanezt a két utasítást, de a **CIRCLE** végére tegyünk három vesszőt és egy kettest. Ez ellipszist csinál. Annak érdekében, hogy ne kelljen félórát várni az extrák megjelenésére, **ujelem**ben **'** mögé tettem a **queue**-nak való értékadásokat és mögéjük odamásoltam a sorokat ugyanúgy, de **elemek + 1** helyett **2 + 11**-et írva. Így kizárólag **V** elemeket és extrákat kapunk.

A dolog működik, tényleg kétféle külsejű extra érkezik, nagyjából fele-fele arányban. Csináljuk meg, hogy kétféleképpen is működjenek.

Ez az **extra** rutin hatásköre. Tegyünk ebbe is egy **SELECT CASE ext**-et, és a rutin eddigi szövege kerüljön a **CASE 1** ágba. Illetve az **IF ELSE** ága változatlanul marad, mert az **IF** azt vizsgálja, hogy az extrát az akna fenekénél magasabban dobtuk-e le. A fenékre dobva minden extránk ugyanúgy viselkedik (nem csinál semmit), tehát először vizsgáljuk meg, hogy hol vagyunk, s csak aztán az extra típusát. Tehát a **SELECT CASE** 4. sornak jön, utána a **CASE 1**, majd az **IF** igaz ága, a **CASE 2**, az **END SELECT**, és ezután a teljes **ELSE** ág meg ami hátravan.

Írjuk meg a **CASE 2** ágat. Tulajdonképpen ugyanúgy működik, mint a **CASE 1**, csak ciklusba téve: addig ismétli a kockakiütést, amíg maga alatt kockát lát. A kockakiütést a **CASE 1** ág első három sora végzi, ezt tegyük egy **DO … LOOP** ciklusba. Legyen hátultesztelő, mert először mindenképpen kiütünk egy kockát, csak utána nézzük meg, hogy van-e több kocka is. Hogyan? Növeljük eggyel **ey**-t, aztán nézzük meg, hogy az új aknahelyen van-e kocka. Ha van, ismétlünk: **LOOP WHILE akna(ex, ey).szin**. Aztán jöhet a **CASE 1** ág maradék három sora, amik rendet tesznek az extra után.

Nos, a dolog nem tökéletes, mert a kettes extra is csak egy kockát cserél ki. Mi van, ha a **WHILE**-t kicseréljük **UNTIL**-ra, ellenkezőjére fordítva a feltételt?

Akkor **Subscript out of range**-et kapunk, mert **ey** eléri az akna fenekét, és ott sem áll meg.

Persze! Nem az **ey**. sort kell vizsgálni, mert abban az extra van, hanem az utána következőt. Tegyük csak vissza a **WHILE**-t, és tegyünk bele **ey + 1**-et.

Az extránk csodálatosan működik, hatalmas oszlopokat egyetlen mozdulattal eltüntet, de szigorúan csak a legelső lyukig.

Jó munkát végeztünk, de a problémát nem oldottuk meg. A második pálya gyakorlatilag megnyerhetetlen. Lehet, hogy a megoldás a pálya áttervezése lesz, de egyelőre ez a pálya nagyszerűen használható arra, hogy a nagy kockahalmokkal való küzdelem speciális eseteit gyakoroljuk.

Mi lenne, ha bővítenénk az arzenált? Létrejöhetne egy harmadik fajta bomba, ami máshogy rombol. Mondjuk egy nagy tömbből kirobbant egy szabálytalan darabot.

A következőképpen képzelem el a dolgot. A bombát rádobjuk egy olyan kockára, amit balról, jobbról és alulról is kockák határolnak. A bomba feljegyzi ennek a három szomszédos kockának a koordinátáit, aztán mind a hármat megvizsgálja sorban. Feljegyzi a szomszédaik koordinátáit. Ezeknek a szomszédainak is feljegyzi a koordinátáit, és így tovább. Ha minden kockát eltüntetne, aminek ily módon feljegyezte a koordinátáit, akkor eltűnne az egész óriási halom, aminek egyetlen kockájára rádobtuk a bombát – ez se rossz, de talán túlzás. Csináljuk azt, hogy a bomba egy kezdeti értékkel induljon, és minden koordinátapár feljegyzésekor csökkentse ezt az értéket. Ha nullára csökken, akkor a bomba nem jegyez föl több koordinátapárt.

Próbáljuk ki, mi lesz. Növeljük meg **dbext**-et háromra, **extrarajz**ban pedig helyezzünk el egy újabb ágat:

```
CASE 3
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , 2
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , .5
PAINT (px + kocka / 2, py + kocka / 2), 8
```

Némi kísérletezés után választottam ezt a módszert, ez egy egész helyes kis piros keresztet eredményez, és a mindenfelé való hatás jelképének tekinthető.

Jöhet az **extra**, itt is kell egy új ág, és csináljuk meg a koordinátajegyezgetést. A dolog működjön úgy, mint a **queue**: egy szöveges változóba rakosgassuk a koordinátapárokat, két karakter jelképez egy kockát. Először állítsuk be azt a bizonyos energiaértéket, legyen a neve **erg**, és legyen benne jó nagy szám, például 100. Később majd csökkentjük, de most lássuk, hogyan irtja a kockákat.

A szövegváltozó neve legyen **ko$** a koordináták alapján. Először tegyük bele a bombával közvetlenül megtámadott kocka koordinátáit, aztán vizsgáljuk meg a szomszédokat.

```
CASE 3: erg = 100: kockatorles ex, ey: ey = ey + 1
ko$ = CHR$(ex) + CHR$(ey)
DO: v = 0
```

IF akna(ex - 1, ey).szin THEN ko$ = ko$ + CHR$(ex - 1) + CHR$(ey): v = 1
IF akna(ex + 1, ey).szin THEN ko$ = ko$ + CHR$(ex + 1) + CHR$(ey): v = 1
IF akna(ex, ey - 1).szin THEN ko$ = ko$ + CHR$(ex) + CHR$(ey - 1): v = 1
IF akna(ex, ey + 1).szin THEN ko$ = ko$ + CHR$(ex) + CHR$(ey + 1): v = 1

Magát a bombát tehát rögtön az elején töröltem, és **ey**-t átállítottam a bombáról a kockára. A bomba négy irányba hat, fölfelé is; az első kockának nem lesz felső szomszédja, hiszen ott a bomba volt, de a későbbieknek lehet. A koordinátafeljegyzést természetesen ciklusban végezzük, és mielőtt bármit fölírnánk, egy **v** változóban megjegyezzük, hogy nem volt felírnivaló; ha valamit felírunk, akkor jelezzük, hogy megtettük.

Ezután csökkentsük eggyel **erg**et, és zárjuk be a ciklust: addig kell futnia, amíg **v** nulla nem lesz (vagyis nem volt új kocka, amit föl kellett volna írni) vagy **erg** el nem fogy. **LOOP UNTIL v = 0 OR erg = 0**.

Bocsánat, ez így nem jó. Ha teszünk mögé egy **STOP**-ot és kiíratjuk **ko$**-t, pláne file-ba az *Immediate* ablak segítségével, akkor kiderül, hogy igencsak szegényes az a koordinátakészlet, ami belekerült. Persze, mert **ex**-et és **ey**-t nem változtattuk meg, örökké csak a kezdő kocka szomszédait írogatja föl. Változtassuk meg. Miután egy kocka szomszédait fölírta, vegye az első szomszédot, írja föl annak a szomszédait, aztán a következőt, s így tovább. Hogy lehet ezt megcsinálni?

Vegyünk egy mutatót: egy változót, ami a legközelebb megvizsgálandó kocka koordinátáinak **ko$**-beli pozícióját tartalmazza. Kezdetben 3 lesz, mert az első két karakter a kezdő kockát kódolja, azt nem szabad újra megnézni, aztán a mutató által jelzett koordinátákat dekódoljuk **ex**-be és **ey**-ba, a mutatót pedig a következő kocka kódjára állítjuk:

ex = ASC(MID$(ko$, m)): ey = ASC(MID$(ko$, m + 1))
m = m + 2

A **ko$**-nak való kezdeti értékadás mögé pedig **m = 3** kerül.

Ko$ tartalmának újabb vizsgálata azt mutatja, hogy kicsit már változékonyabbak a számok, de túl sok az ismétlés. Ne tegyük bele ugyanazt a koordinátapárt kétszer, háromszor, ötvenszer. Így **erg** túl gyorsan elfogy, mi pedig időt és memóriát pazarlunk. Az **IF** utasításoknak ez már túl bonyolult feladat; cseréljük őket ki rutinhívásokra.

extra3ko ko$, ex - 1, ey, v
extra3ko ko$, ex + 1, ey, v
extra3ko ko$, ex, ey - 1, v
extra3ko ko$, ex, ey + 1, v

Így neveztem el a rutint, mert a „koordinátavizsgálat” vagy „szomszédok” túl általános lett volna, ez tulajdonképpen egy rutinhoz készített „alrutin”, utaljon erre a neve.

SUB extra3ko (ko$, x, y, v)
IF akna(x, y).szin THEN
c$ = CHR$(x) + CHR$(y)
i = INSTR(ko$, c$)

```
IF i / 2 = i \ 2 THEN ko$ = ko$ + c$: v = 1
END IF
END SUB
```

Így fest maga a rutin. Ha a megadott koordinátákon kocka van, bekódolja azokat **c$**-ba, majd megkeresi, hogy szerepel-e **ko$**-ban. Ha **i** páros szám, akkor nem szerepel, mert a nulla azt jelenti, hogy egyáltalán nem található meg, a többi páros szám pedig azt, hogy egy kocka **y** koordinátája és a következő kocka **x** koordinátája egymás mögé írva ugyanaz, mint a mi kockánk két koordinátája, de ez persze nem valóságos találat. A módszer egyébként nem egészen korrekt, mert egy ilyen „felemás találat” mögött ott állhat a valódi koordinátapár, de ez hátha nem fog túl sűrűn előfordulni.

A szűrés olyan jól sikerült, hogy **ko$** csak 24 karakter hosszú lett, azaz tizenkét kockát tartalmazott még azután is, hogy egy jókora emelvényre dobtam rá a „keresztbombát”, sőt kiiktattam **erg** csökkentését. Ez szomorú, de most már csináljuk meg magát a kockák eltávolítását is, akkor látszik majd, hogy melyik kockákat is pécézi ki voltaképpen. Vegyük ki tehát a **STOP**-ot (**erg** csökkentését vissza lehet tenni), és tüntessünk el minden kockát, aminek a koordinátái megvannak **ko$**-ban.

```
FOR m = 1 TO LEN(ko$) STEP 2
x = ASC(MID$(ko$, m)): y = ASC(MID$(ko$, m + 1))
akna(x, y).szin = 0
kockatorles x, y
NEXT
aknarajz
racs
pontozas 0
```

Egy nagyobbacska téglahalomból szépen kiüt egy darabot, ezzel nincs gond, csak jobb lenne, ha akkora darabot ütne ki, amekkorát mi szeretnénk. **Erg** figyelésének kiiktatása nem segít. Mi lenne, ha ehelyett **v** figyelését iktatnánk ki?

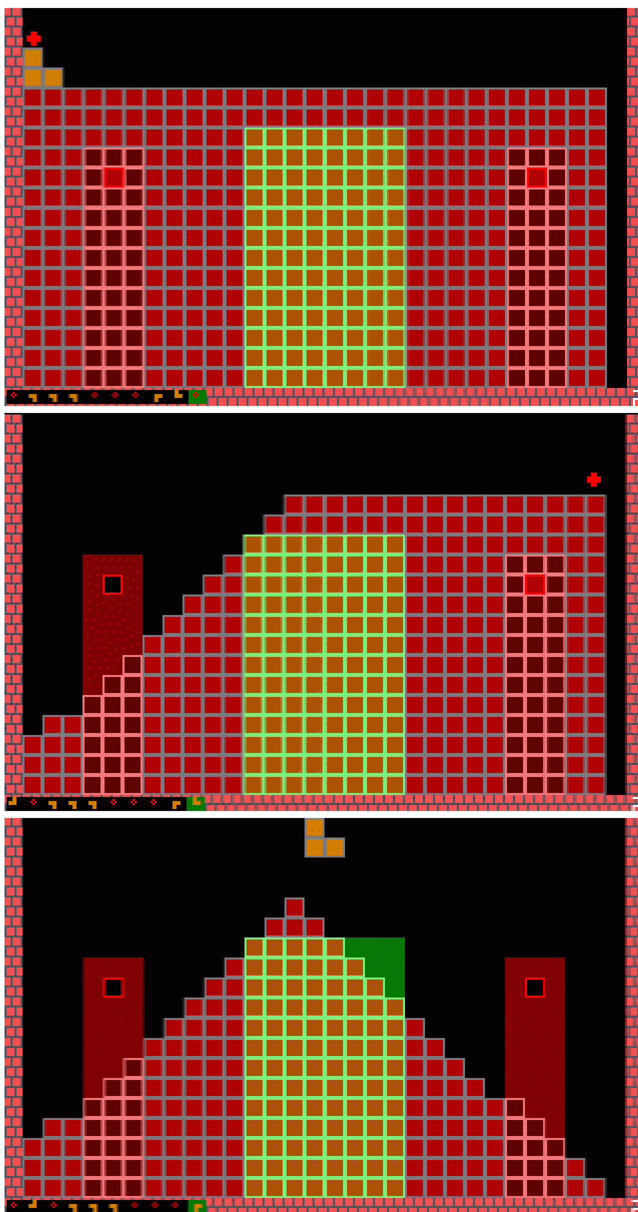
Az lett, hogy az első keresztbomba ledobásakor **Illegal function call** érkezett onnan, hogy **ex**-be tesszük a következő kocka *x* koordinátáját. **M** értéke 63, **ko$** viszont csak 62 karakter hosszú. Esetleg **v** figyelése helyett tegyük be, hogy **m** nem lehet nagyobb **ko$** hosszánál.

A helyzet kicsit javult. A bomba hatalmas építményeket képes eltüntetni, máskor viszont csak egy kilógó nyúlványt pusztít el. Ha pedig történetesen magányos kockára dobjuk rá, **Illegal function call**t ad. Utóbbit nyilván azért teszi, mert **m**-et rögtön az elején 3-ra állítjuk. Tegyük át az **m > LEN(ko$)** feltételt egy **IF**-be, **erg** csökkentése mögé.

A program csodálatosképpen megjavult, a keresztbomba óriási hatásfokkal működik. Kísérletképpen betettem a **mozgas** elejére egy sort:

```
FOR x = 2 TO 30: FOR y = 5 TO 19: akna(x, y).szin = 16: akna(x, y).fal = 0:
NEXT: NEXT: aknarajz
```

Íme három kép a keresztbomba bevetése előtt és kétszeri bevetése után:



Amint látható, a bomba iszonyú kockahalmokat képes elpusztítani. Persze **erg** kezdő értéke lehet akár ezer is, s akkor ebből az egész aknát kitöltő halomból alig marad valami. Nagyon hatékony fegyvert fejlesztettünk ki.

A **v** változó ilyenformán fölöslegessé vált, kidobhatjuk **extrá**ból, és **extra3ko** paraméterei között sem kell többé szerepelnie.

Attól nem kell tartani, hogy a második pálya ezáltal túl könnyen megnyerhetővé válik. Az első próbajáték során csak akkor jött végre egy keresztbomba, amikor már alig volt mozgástér a pálya közepén, s a bomba csak arra adott lehetőséget, hogy egy kicsit előbbről kezdjem a pálya betöltését. Ez persze nem baj. A tetris logikai játék, de nem olyan, mint a sakk, óriási szerepe van benne a véletlennek. Ha legalább egyszer, egyvalakinek sikerül megnyerni a játékot elejétől végéig, akkor a játék megnyerhető.

Erről jut eszembe, érdemes lenne dokumentálni, hogyan folyt le egy-egy játszma tetrisünkben. Esetleg a következő lépést ebbe az irányba tehetnénk.

```
DECLARE SUB extra3ko (ko$, x%, y%)
DECLARE SUB extrarajz (x%, y%)
DECLARE FUNCTION filter% (f$, x%, y%)
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB mozgas ()
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
szin AS INTEGER
tolt AS INTEGER
fal AS INTEGER
kapcs AS INTEGER
END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10
CONST dbext = 3

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
```

```
DIM SHARED ply, pont, zus, zuk, ext
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
DIM SHARED par(1 TO 10) AS STRING, mpar

inic
aknarajz
mozgas

DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
kockarajz x, y, 256, 0
ELSE
IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": extrarajz x, y
END SELECT
NEXT
```

```
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
kockatorles x, y
IF akna(x, y).tolt THEN racsrajz x, y
IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "COPY": x1 = VAL(par(1)): y1 = VAL(par(2))
x2 = VAL(par(3)): y2 = VAL(par(4))
xn = VAL(par(5)): yn = VAL(par(6))
IF x1 > x2 THEN SWAP x1, x2
IF y1 > y2 THEN SWAP y1, y2
FOR x = 0 TO x2 - x1: FOR y = 0 TO y2 - y1
IF filter(par(7), x1 + x, y1 + y) THEN
akna(xn + x, yn + y) = akna(x1 + x, y1 + y)
kockatorles xn + x, yn + y
END IF
NEXT: NEXT
CASE "FILL": x1 = VAL(par(2)): y1 = VAL(par(3))
x2 = VAL(par(4)): y2 = VAL(par(5))
IF x1 > x2 THEN SWAP x1, x2
IF y1 > y2 THEN SWAP y1, y2
FOR xx = 0 TO x2 - x1: FOR yy = 0 TO y2 - y1
x = x1 + xx: y = y1 + yy
IF filter(par(6), x, y) THEN
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END IF
NEXT: NEXT
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
SELECT CASE VAL(par(1))
CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y
```

```
CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
kockatorles x, y: racsrajz x, y
CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
kockatorles x, y: racsrajz x, y
CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, 256, 0
CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
END SELECT
END SELECT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
SELECT CASE ext
CASE 1
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
aknarajz
racs
pontozas 0
CASE 2
DO
akna(ex, ey + 1).szin = 0
kockatorles ex, ey
kockatorles ex, ey + 1
ey = ey + 1
LOOP WHILE akna(ex, ey + 1).szin
aknarajz
racs
pontozas 0
CASE 3: erg = 100: kockatorles ex, ey: ey = ey + 1
ko$ = CHR$(ex) + CHR$(ey): m = 3
DO
extra3ko ko$, ex - 1, ey
extra3ko ko$, ex + 1, ey
extra3ko ko$, ex, ey - 1
extra3ko ko$, ex, ey + 1
erg = erg - 1
IF m > LEN(ko$) THEN EXIT DO
ex = ASC(MID$(ko$, m)): ey = ASC(MID$(ko$, m + 1))
m = m + 2
LOOP UNTIL erg = 0

FOR m = 1 TO LEN(ko$) STEP 2
x = ASC(MID$(ko$, m)): y = ASC(MID$(ko$, m + 1))
akna(x, y).szin = 0
kockatorles x, y
NEXT
aknarajz
racs
pontozas 0
END SELECT
ELSE
kockatorles ex, ey
racsrajz ex, ey
END IF
```

```
END SUB

SUB extra3ko (ko$, x, y)
IF akna(x, y).szin THEN
c$ = CHR$(x) + CHR$(y)
i = INSTR(ko$, c$)
IF i / 2 = i \ 2 THEN ko$ = ko$ + c$
END IF
END SUB

SUB extrarajz (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE ext
CASE 1
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE 2
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8, , , 2
PAINT (px + kocka / 2, py + kocka / 2), 8
CASE 3
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , 2
CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , .5
PAINT (px + kocka / 2, py + kocka / 2), 8
END SELECT
END SUB

FUNCTION filter (f$, xx, yy)
x = xx: y = yy
FOR a = 1 TO LEN(f$)
x$ = MID$(f$, a, 1)
SELECT CASE x$
CASE "x": x = x - 1
CASE "X": x = x + 1
CASE ".": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "*": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "/": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "#": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 1 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "B", "b": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "G", "g": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "R", "r": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).fal = 1
akna(aknax, y).fal = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).fal = 1
```

```
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
ply = 1
palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
```

```
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
LINE (px, py)-STEP(3, 3), 6, BF
LINE (px + 5, py)-STEP(3, 3), 6, BF
LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
LINE (px, py + 5)-STEP(0, 3), 6
LINE (px, py + 4)-STEP(9, 0), 7
LINE (px, py + 9)-STEP(9, 0), 7
LINE (px + 4, py)-STEP(0, 3), 7
LINE (px + 9, py)-STEP(0, 3), 7
LINE (px + 1, py + 5)-STEP(0, 3), 7
LINE (px + 6, py + 5)-STEP(0, 3), 7
CASE ELSE
LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
END SELECT
END SUB

SUB kockatorles (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 0, BF
END SUB

FUNCTION mehet (e, i, ex, ey)
```

```
x = ex: y = ey: mehet = 1
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB mozgas
ujelem e, i, x, y
DO
elemrajz e, i, x, y
a$ = INKEY$
SELECT CASE a$
CASE CHR$(0) + "K"
IF mehet(e, i, x - 1, y) = 1 THEN
elemtorles e, i, x, y: x = x - 1
ELSE
IF raktar(e, i) = "x" AND akna(x - 1, y).fal = 0 THEN
akna(x - 1, y).szin = 0
kockatorles x - 1, y
kockatorles x, y
racsrajz x - 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
CASE CHR$(0) + "M"
IF mehet(e, i, x + 1, y) = 1 THEN
elemtorles e, i, x, y: x = x + 1
ELSE
IF raktar(e, i) = "x" AND akna(x + 1, y).fal = 0 THEN
akna(x + 1, y).szin = 0
kockatorles x + 1, y
kockatorles x, y
racsrajz x + 1, y
racsrajz x, y
ujelem e, i, x, y
END IF
END IF
CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
SELECT CASE mehet(e, ii, x, y)
CASE 1: elemtorles e, i, x, y: i = ii
CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
CASE 4
IF mehet(e, ii, x - 1, y) = 1 THEN
elemtorles e, i, x, y: i = ii: x = x - 1
ELSE
IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
END IF
CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
```

```
END SELECT
CASE CHR$(0) + "P"
IF mehet(e, i, x, y + 1) = 1 THEN
elemtorles e, i, x, y: y = y + 1
ELSE ragadas e, i, x, y: ujelem e, i, x, y
END IF
END SELECT
LOOP
END SUB

SUB paletta (p, r, g, B)
OUT &H3C8, p
OUT &H3C9, r
OUT &H3C9, g
OUT &H3C9, B
END SUB

SUB palya (ply)
OPEN "tetris.lev" FOR INPUT AS 2
FOR a = 0 TO 9: prg(a) = "": NEXT
DO
LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 THEN
FOR y = 1 TO aknay - 1
LINE INPUT #2, l$
FOR x = 2 TO aknax - 1
x$ = MID$(l$, x - 1, 1)
IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
akna(x, y).tolt = 0: akna(x, y).fal = 0: akna(x, y).kapcs = 0
SELECT CASE x$
CASE "*": akna(x, y).tolt = 1
CASE "/": akna(x, y).tolt = 2
CASE "#": akna(x, y).fal = 1: akna(x, y).szin = 0
CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
END SELECT
NEXT: NEXT

DO UNTIL EOF(2)
LINE INPUT #2, l$
IF MID$(l$, 2, 1) = ":" THEN
i = VAL(l$)
prg(i) = LTRIM$(MID$(l$, 3))
ELSE
IF VAL(l$) THEN EXIT DO
prg(i) = prg(i) + ":" + l$
END IF
LOOP
CLOSE 2
EXIT SUB
END IF
LOOP UNTIL EOF(2)
CLOSE 2
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
```

```
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
akna(x, y).szin = 0
kockatorles x, y
pontozas 10
END IF
NEXT: NEXT

ply = ply + 1
palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
END IF
END SUB

SUB ragadas (e, i, ex, ey)
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
```

```
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2: vk = 0
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0: vk = 0 ELSE vk = 1
CASE 1
IF tele = 1 THEN
IF akna(x - 1, y).fal = 0 THEN IF vk THEN zuditas y, k, x - 1: k = x + 1
ELSE tele = 1: k = x + 1: vk = 0
END IF
END SELECT
NEXT
IF tele THEN IF vk THEN zuditas y, k, x - 1
NEXT
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
'queue = queue + CHR$(INT(RND * 2 + 11)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
'queue = queue + CHR$(INT(RND * 2 + 11)) + CHR$(INT(RND * 4 + 1))

LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
```

```
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT

x = aknax \ 2
y = 2
ext = INT(RND * dbext) + 1
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)

csf = 1
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1 ELSE csf = 0
IF f(x) = 0 THEN
akna(x, yy).szin = akna(x, yy - 1).szin
kockatorles x, yy
END IF
NEXT: NEXT

FOR x = k TO v
IF f(x) = 0 THEN
akna(x, 1).szin = 0
kockatorles x, 1
END IF
NEXT
aknarajz
IF csf = 0 THEN sorok
END SUB
```

Kamera indul!

Ha a játék egész menetét rögzítjük, akkor azt később akárhányszor vissza lehet játszani. Nagyon érdekes és tanulságos lehet. Nézzük át, mit kell feljegyeznünk.

A pálya adatait (nem elég az, hogy hányadik pálya, mert aztán átszerkesztjük és nem lesz többé érvényes az egész, tehát a teljes pályaleírás kelleni fog).

A **queue** kezdeti állapotát, hogy ugyanúgy rajzolhassuk ki.

Ext értékét.

Az elem teljes útvonalát, tehát minden egyes billentyűlenyomást.

A következőként kisorsolt és a **queue** végére illesztett elemet.

Oké, vágjunk bele. Legyen egy **tetris.flm** nevű file-unk, ami a játékról készített filmet tartalmazza. És legyen egy rutin, ami szépen följegyez bele mindent. Hívjuk **film**nek. Legyen neki egy paramétere, ami megmondja, hogy éppen mi történik:

0 – új pályát kezdtünk, a pálya adatait kell kiírni;

*1 – **queue**-t kell kiírni;*

*2 – **ext**et kell kiírni;*

3 – az elem mozgását kell kiírni;

4 – az újonnan kisorsolt elemet kell kiírni.

A rutin persze elágazik a paraméter értéke szerint, és eldönti, mit írjon ki. Egy fél pillanat, kicsit mégis csináljuk másképpen. A **palya** rutin némi bíbelődéssel találja meg **tetris.lev**ben a szükséges pályát, ezt ne csináljuk már meg kétszer. A **palya** írja ki beolvasáskor a pályaadatokat.

A 9. sorban olvassuk be a pályarajz egy sorát. Ezt egyből át is másolhatjuk a filmbe: **PRINT #3, I$**, és persze kell egy **OPEN "tetris.flm" FOR OUTPUT AS 3** a rutin elejére.

A 25. sorban a programkód egy sorát olvassuk be, ide is betehetjük az előbbi **PRINT**-et. A két **CLOSE** utasítást pedig kicserélhetjük **CLOSE 2, 3**-ra, és ez a része a dolgoknak készen is van.

A **tetris.flm** egy próbajáték után, ahol a második pálya kezdetéig játszottam, nem tartalmazott többet, mint a második pálya rajzát és egy hármast. Persze, mert új pálya kezdetekor felülírjuk a file-t. Cseréljük ki az **OUTPUT**-ot **APPEND**-re és töröljük a file-t.

Így már az első pálya teljes leírása is megvan a file-ban, ámbár a 3-as szám még mindig ott van a végén. A 26. sorbeli **PRINT**-et áttehetjük az **IF**-be, az igaz és a hamis ág végére, így a vizsgálat után írunk csak.

Most már rendben van, leszámítva, hogy a két pályaleírás minden külön jelzés nélkül követi egymást, de majd úgyis ott lesz köztük a pályákon történő események leírása.

Írjuk ki **queue**-t. Ezt már csinálhatja a **film** rutin, hívjuk meg **ujelem** 6. sorából **film 1** utasítással. A rutin ilyenkor kiírja **queue**-t a filmfile-ba, de jobb, ha nem egyszerűen **PRINT**-tel, hanem visszakódolva számokká, hogy a file szövegesen olvasható maradjon.

```
SUB film (f)
OPEN "tetris.flm" FOR APPEND AS 3
SELECT CASE f
CASE 1
PRINT #3, "Q";
FOR a = 1 TO LEN(queue)
PRINT #3, STR$(ASC(MID$(queue, a)));
```

```
NEXT
PRINT #3,
END SELECT
CLOSE 3
END SUB
```

Így ni, a karakterkódokat számokká alakítottuk és egy-egy szóközzel elválasztva írjuk ki egy sorban. A sor elejére egy `Q` betűt tettünk, hogy jelezze, ez a sor a **queue** tartalma.

Ha azonban a **film 1** utasítást áttesszük **ujelem** 6. sorából a 11-edikbe, akkor a **film 4** utasításra már nincs is szükség, a **queue** minden változtatásakor a teljes sort kiírjuk. Ez egyszerűbb lesz.

Nézzük a **film 2** esetet: **ext** kiírását.

```
CASE 2: PRINT #3, "X"; STR$(ext)
```

A kód ennyivel elintézhető, és ezt is **ujelem**ből kell hívni, a legvégéről.

Lássuk a **film 3**-at. Ezt kétféleképpen lehet: minden billentyűlenyomásnál vagy az elem elhelyezése után, utólag. Ha minden billentyűlenyomásnál meghívjuk, akkor a billentyűt is át kell adni **film**nek, és ugyanannak a billentyűnek a többszöri megnyomása több egyforma jelet eredményez; próbáljuk inkább utólag. Tehát **mozgas**nak naplóznia kell, hogy milyen billentyűt nyomtunk meg és hányszor; ha a korábban nyomogatott billentyűtől eltérőt nyomunk meg, akkor jöhet a **film 3**.

Legyen két változónk (globálisak, hogy **film** is elérje őket), **billk** tartalmazza a billentyű kódját, **billn** pedig azt, hogy hányszor nyomtuk le. **Mozgas** 7. sorától, az **INKEY$** után először is számkóddá alakítjuk a billentyűt:

```
IF LEN(a$) = 2 THEN
k = INSTR("KMHP", RIGHT$(a$, 1))
```

majd (ha olyan billentyű volt megnyomva, ami csinál valamit) megnézzük, hogy előzőleg is ezt a billentyűt nyomták-e le; ha igen, akkor **billn**-t eggyel növeljük, ellenkező esetben kiírjuk az eddigieket és **billn**-t nullázzuk. Ezután **billk**-ba bemásoljuk **k**-t.

```
IF k THEN
IF k = billk THEN billn = billn + 1 ELSE film 3: billn = 0
billk = k
END IF
END IF
```

A **film 3**-nak az a dolga, hogy kiírja **billk** és **billn** értékét:

```
CASE 3: PRINT #3, "M"; STR$(billk); STR$(billn)
```

A gond az, hogy `M 0 0` is megjelenik a filmfile-ban. Egy **IF billk = 0 THEN STOP** elhelyezése a **film**ben megmutatja, hogy ez az első gombnyomásnál történik. Nyilván azért, mert bármi is az elsőként megnyomott gomb, a kódja nem egyezhetik meg a semmivel, ami a változóban kezdetként van. Ezért **film** csak akkor írjon, ha **billk** nem 0.

Valami még mindig nem stimmel. Csináltam egy próbát: egyetlen függőleges mozdulattal letettem egy elemet. A filmbe nem került `M` bejegyzés. Másik próba: háromszor elfordítom az elemet, aztán leteszem. `M 3 2` jelenik meg a file-ban, ami a legnagyobb jóakarattal is csak kétszeri elfordítást jelent, és a letevésének nincs nyoma, `Q` és `X` bejegyzés mutatja, hogy új elemet sorsoltunk. Harmadik próba: függőlegesen leviszem az elemet a padlóig, egyet lépek vele balra, aztán leteszem. Eredmény: `M 4 16` és `M 1 0`.

Csakugyan: **film** meghívása előtt **mozgas** nem növeli **billn**-t, ezért mindig eggyel kisebb számot kapunk a file-ban. Tegyük át **billn** növelését egy sorral följebb, és mivel így nem marad igaz ág, fordítsuk meg a 11. sorbeli feltételt. Nézzük így.

Függőlegesen levitt elem: nincs **M** bejegyzés. Három elfordítás után lerakás: `M 3 3`. Függőlegesen le padlóig, egyet balra, le: `M 4 17` és `M 1 1`.

A számok tehát már jók, de az *utolsó* mozdulat, az elem lerakása mindig kimarad. Persze, mert ha az elem lerakható, akkor **mozgas** azon nyomban meghívja **ragadas**t és **ujelem**et, nem törődve a mi filmezésünkkel. Tegyük be az 58. sorba **ragadas** elé, hogy **billn = billn + 1: film 3**.

Úgy néz ki, most már minden jó, de ha leteszünk két elemet és megnézzük a filmet, kiderül, hogy mégse jó. Először letettem egy elemet (`M 4 18`), aztán egyet megforgattam a levegőben, és a forgatást jelző bejegyzés előtt kaptam egy `M 4 19`-et. Lerakás után célszerű lesz **billk**-t nullázni.

Valószínűleg most lesz jó – majd ha megcsináljuk a visszajátszást, kiderül, hogy tényleg jó-e. De mielőtt ehhez hozzálátunk, különítsük el a filmbeli pályarajzot a többi szövegtől, mert így képtelenség lesz feldolgozni. Mondjuk **palya** végére, a file-ok bezárása után tehetünk egy **film 0** utasítást, ez a szám úgyis gazdátlanul maradt, és jelentse azt, hogy végjel gyanánt egy 0-t írjunk ki külön sorba. Elvégre **palya** úgyis addig olvassa a file-t, amíg egy számot nem talál önállóan egy sorban.

Ezzel a rendszerrel elkészítettem egy filmfile-t, ami eddigi három pályánk megnyerését tartalmazza:

```
..............................
..............................
..............................
..............................
..............................
..............................
..........#********#..........
...///....#********#....///...
.../1/....#********#..../2/...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
1: put 0 12 7:put 0 12 8:put 0 12 9:put 0 12 10:put 0 12 11
put 0 12 12:put 0 12 13:put 0 12 14:put 0 12 15:put 0 12 16
put 0 12 17:put 0 12 18:put 0 12 19
2: put 3 12 7:put 3 12 8:put 3 12 9:put 3 12 10:put 3 12 11
put 3 12 12:put 3 12 13:put 3 12 14:put 3 12 15:put 3 12 16
put 3 12 17:put 3 12 18:put 3 12 19
0
Q 2 1 7 4 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2
X 3
M 4 8
M 2 5
M 4 10
Q 7 4 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1
X 1
M 1 12
M 3 1
M 4 16
Q 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4
X 2
M 1 13
M 4 7
M 2 1
M 4 6
Q 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2
X 1
M 1 12
M 4 5
M 3 2
M 4 5
Q 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4
X 2
M 4 12
M 2 1
M 4 5
Q 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4
X 1
M 1 11
M 4 8
Q 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3
X 2
```

M 4 13
M 3 1
M 1 1
M 4 5
Q 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3
X 1
M 2 3
M 4 16
Q 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4
X 1
M 1 2
M 4 17
Q 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4
X 2
M 3 1
M 4 16
Q 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2
X 1
M 1 11
M 4 7
Q 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1
X 2
M 1 1
M 4 9
M 2 1
M 4 5
Q 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1
X 3
M 3 1
M 2 3
M 4 14
M 1 1
M 4 1
Q 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4
X 2
M 4 8
M 2 2
M 4 5
Q 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2
X 2
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 11
Q 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2
X 1
M 3 3
M 4 13
Q 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2
X 3
M 1 3
M 4 17
Q 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2
X 1
M 3 1

M 1 2
M 4 15
Q 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2
X 2
M 1 4
M 4 5
M 2 1
M 4 9
Q 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2
X 1
M 4 6
M 1 2
M 4 6
Q 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1
X 3
M 1 3
M 4 10
Q 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4
X 3
M 4 11
Q 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3
X 3
M 3 2
M 2 4
M 4 12
Q 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3
X 1
M 4 1
M 1 2
M 4 9
Q 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1
X 1
M 2 2
M 3 2
M 1 1
M 4 9
Q 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2
X 3
M 2 1
M 3 1
M 2 2
M 4 11
Q 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2
X 3
M 2 3
M 3 3
M 4 10
Q 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2
X 3
M 4 8
Q 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1
X 1
M 2 2
M 4 7
Q 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1
X 2
M 2 1
M 1 2
M 3 2

```
M 1 2
M 4 7
Q 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4
X 2
M 3 1
M 1 2
M 4 7
Q 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2
X 3
M 2 7
M 4 13
M 1 1
M 4 4
Q 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2
X 3
M 1 12
M 4 2
M 1 1
M 4 1
M 2 1
M 4 4
Q 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4
X 2
M 3 1
M 2 3
M 4 8
Q 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3
X 3
M 3 1
M 1 1
M 4 7
Q 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3
X 2
M 2 3
M 4 6
Q 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2
X 1
M 2 1
M 4 5
Q 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1
X 1
M 1 4
M 4 5
Q 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3
X 2
M 2 3
M 3 1
M 2 2
M 4 1
M 1 1
M 4 4
..............................
..............................
..............................
..............................
########/............/########
********/............/********
********/............/********
********/............/********
```

```
********/............/********
********/............/********
########/............/########
..............................
..............................
.............####.............
.............****.............
.............****.............
/////////////****/////////////
/////////////****/////////////
/////////////****/////////////
0
Q 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1
X 3
M 1 3
M 4 5
M 1 9
M 4 3
M 2 2
M 4 1
Q 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3
X 1
M 3 1
M 1 4
M 4 5
M 1 11
M 4 4
Q 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1
X 2
M 1 4
M 4 5
M 1 8
M 4 3
Q 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2
X 1
M 3 1
M 4 1
M 1 2
M 4 3
M 1 1
M 4 1
M 1 5
M 4 4
Q 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3
X 3
M 4 6
M 3 1
M 2 14
M 4 3
Q 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3
X 1
M 4 1
M 1 1
M 4 4
M 2 16
M 4 2
Q 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2
X 1
M 1 5
```

```
M 4 4
M 1 8
M 4 4
Q 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2
X 3
M 1 2
M 4 9
M 1 1
M 3 1
M 1 1
M 4 3
M 2 1
M 4 1
M 2 4
M 4 5
Q 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2
X 3
M 1 1
M 4 7
M 2 11
M 4 2
Q 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1
X 1
M 1 3
M 4 3
M 1 1
M 4 2
M 1 4
M 4 2
Q 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2
X 1
M 1 5
M 4 4
M 1 11
M 4 3
Q 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2
X 2
M 1 1
M 4 7
M 1 2
M 4 7
M 2 2
M 4 2
Q 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1
X 3
M 4 8
M 2 7
M 4 8
Q 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1
X 3
M 1 2
M 3 1
M 4 5
M 2 10
M 4 1
M 2 2
M 1 1
M 4 2
Q 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3
```

```
X 2
M 1 5
M 4 4
M 1 9
M 4 2
Q 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1
X 2
M 1 2
M 4 7
M 1 1
M 4 6
M 2 4
M 3 1
M 4 1
M 3 1
M 4 2
Q 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3
X 2
M 1 3
M 3 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 5
M 4 3
Q 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2
X 2
M 1 2
M 4 5
M 2 15
M 4 3
Q 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2
X 1
M 1 1
M 4 5
M 2 13
M 4 2
Q 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4
X 1
M 2 1
M 4 4
Q 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4
X 1
M 2 2
M 4 4
Q 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4
X 3
M 3 1
M 1 3
M 4 3
M 1 1
M 4 1
M 1 11
M 4 1
Q 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3
```

```
X 2
M 1 4
M 4 4
M 1 9
M 4 1
Q 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4
X 2
M 3 1
M 4 6
M 2 3
M 4 1
M 2 1
M 4 7
M 1 2
M 4 2
Q 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1
X 1
M 4 5
M 2 14
M 4 1
Q 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1
X 1
M 2 3
M 3 1
M 2 3
M 4 3
M 2 1
M 4 1
M 3 1
M 2 9
M 4 1
Q 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4
X 1
M 1 3
M 4 14
M 2 3
M 4 1
Q 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4
X 1
M 3 1
M 4 5
M 2 9
M 4 2
Q 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2
X 1
M 4 2
M 2 6
M 4 1
M 2 1
M 4 1
M 2 2
M 1 2
M 4 7
M 1 4
M 4 2
M 1 1
M 4 1
Q 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3
X 3
```

```
M 4 6
M 1 1
M 4 1
M 1 1
M 4 6
M 2 1
M 4 1
Q 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2
X 3
M 3 1
M 4 8
M 2 1
M 4 3
Q 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2
X 2
M 3 2
M 2 6
M 4 4
M 2 7
M 4 1
Q 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4
X 3
M 3 1
M 2 6
M 4 1
M 2 1
M 4 4
M 2 5
M 4 1
Q 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4
X 2
M 3 1
M 1 2
M 4 10
M 2 1
M 4 1
Q 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1
X 2
M 1 3
M 4 4
M 1 7
M 4 2
Q 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3
X 1
M 4 6
M 2 1
M 4 3
Q 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2
X 1
M 1 1
M 3 1
M 1 1
M 4 9
M 2 1
M 4 1
Q 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1
X 1
M 2 5
M 4 4
```

Q 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1
X 1
M 4 2
M 2 1
M 4 1
M 2 1
M 4 5
Q 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2
X 1
M 4 2
M 1 3
M 4 1
M 2 9
M 4 1
M 2 3
M 4 1
Q 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4
X 1
M 1 2
M 4 8
Q 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2
X 3
M 2 6
M 4 7
M 3 1
M 1 2
M 4 4
M 1 1
M 4 1
Q 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1
X 3
M 1 1
M 3 1
M 1 4
M 4 4
M 1 6
M 4 1
Q 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4
X 1
M 2 3
M 4 8
Q 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4
X 1
M 4 1
M 3 1
M 4 6
Q 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4
X 2
M 1 1
M 4 6
Q 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1
X 2
M 3 1
M 2 2
M 4 6
Q 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2
X 2
M 2 3
M 3 1

```
M 2 2
M 4 6
M 1 1
M 4 4
Q 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1
X 1
M 2 3
M 4 4
Q 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2
X 1
M 2 7
M 4 4
M 2 1
M 4 1
..............................
..............................
..............................
..............................
..............................
#######................#######
******#................#******
******#................#******
5*****#................#*****6
******#................#******
******#................#******
******#................#******
#######................#######
******#................#******
******#.......12.......#******
7*****#.......34.......#*****8
******#................#******
******#................#******
******#...9............#******
1: put 2 8 9: put 2 8 10
2: put 2 8 16: put 2 8 17
3: put 2 25 16: put 2 25 17
4: put 2 25 9: put 2 25 10
5: put 3 8 16: put 3 8 17: put 2 8 15: put 2 8 14
6: put 3 25 16: put 3 25 17: put 2 25 15: put 2 25 14
7: put 3 25 9: put 3 25 10: put 2 25 8: put 2 25 7
8: put 3 8 9: put 3 8 10: put 2 8 8: put 2 8 7
9: copy 8 6 8 19 9 6 *bg/r#: fill 0 8 6 8 19 x.*bg/r
copy 25 6 25 19 24 6 *bg/r#: fill 0 25 6 25 19 X.*bg/r
0
Q 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4
X 2
M 1 4
M 3 1
M 4 13
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 1 3
M 4 3
Q 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2
X 3
M 2 5
```

M 4 12
M 3 1
M 4 2
M 1 3
M 4 1
M 1 1
M 4 3
Q 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4
X 2
M 2 6
M 4 11
M 1 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 4
M 4 1
M 1 1
M 4 2
Q 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1
X 2
M 2 7
M 3 1
M 4 13
M 1 4
M 4 1
M 1 1
M 4 3
Q 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3
X 2
M 1 3
M 4 11
M 3 1
M 4 2
M 2 2
M 4 3
Q 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3
X 2
M 1 5
M 3 3
M 4 9
M 2 1
M 4 3
M 2 7
M 1 2
M 4 3
Q 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4
X 3
M 1 3
M 4 13
M 2 1
M 4 3
Q 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4
X 3
M 3 1
M 1 4
M 4 12

```
M 2 5
M 1 1
M 4 2
Q 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2
X 3
M 2 6
M 4 1
M 2 1
M 4 6
M 2 8
M 4 3
Q 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4
X 1
M 3 1
M 1 4
M 4 2
M 1 1
M 4 1
M 1 1
M 4 5
M 1 7
M 4 3
Q 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2
X 2
M 1 5
M 3 1
M 4 15
M 1 11
M 4 3
Q 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4
X 2
M 2 7
M 4 1
M 3 1
M 4 7
M 2 7
M 3 1
M 4 1
Q 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2
X 1
M 2 3
M 3 1
M 2 3
M 4 1
M 2 1
M 4 7
M 2 4
M 4 2
M 2 1
M 4 1
Q 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2
X 1
M 2 5
M 3 1
M 2 1
M 3 1
M 4 1
M 2 1
M 4 6
```

M 2 6
M 4 2
Q 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3
X 3
M 1 5
M 4 1
M 1 1
M 4 6
M 1 6
M 3 2
M 1 1
M 3 1
M 2 2
M 3 2
M 2 5
M 4 7
M 1 7
M 3 3
M 2 2
M 3 1
M 1 1
M 4 3
Q 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4
X 1
M 1 3
M 3 1
M 1 2
M 4 3
M 3 1
M 4 2
M 1 1
M 4 2
M 1 7
M 3 1
M 1 1
M 4 2
Q 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2
X 2
M 1 4
M 3 1
M 1 2
M 4 13
M 1 7
M 3 1
M 1 1
M 4 2
Q 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3
X 1
M 2 6
M 4 11
M 3 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 7
M 4 3
Q 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2

```
X 1
M 1 4
M 4 2
M 1 1
M 4 4
M 3 1
M 4 2
M 1 5
M 4 2
M 1 1
M 4 1
Q 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1
X 3
M 2 7
M 4 1
M 3 1
M 4 14
M 2 5
M 1 1
M 3 1
M 4 1
M 2 1
M 4 1
Q 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4
X 2
M 2 1
M 1 3
M 2 8
M 4 14
M 3 2
M 1 1
M 4 1
M 1 1
M 4 2
Q 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4
X 2
M 1 4
M 4 9
M 3 2
M 4 3
M 2 1
M 4 3
Q 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1
X 3
M 3 1
M 1 3
M 2 9
M 4 2
M 2 1
M 4 1
M 2 1
M 4 2
M 2 9
M 4 2
Q 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2
X 1
M 2 8
M 4 7
M 1 1
```

M 3 1
M 4 6
M 2 6
M 3 1
M 2 1
M 4 3
Q 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3
X 2
M 1 4
M 3 1
M 4 8
M 3 1
M 4 4
M 2 2
M 3 1
M 4 1
Q 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3
X 2
M 3 3
M 2 6
M 4 12
M 1 7
M 4 1
Q 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3
X 1
M 1 1
M 2 1
M 1 3
M 4 13
Q 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4
X 3
M 3 1
M 1 4
M 4 11
Q 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1
X 3
M 1 4
M 3 3
M 1 1
M 4 7
M 1 1
M 4 1
M 1 7
M 4 1
Q 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4
X 1
M 2 6
M 3 1
M 4 1
M 2 1
M 4 11
M 2 11
M 4 2
Q 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1
X 3
M 3 1
M 2 5
M 3 1
M 2 1

M 4 13
M 1 3
M 4 1
Q 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2
X 2
M 2 3
M 4 3
Q 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3
X 2
M 1 6
M 4 15
M 2 1
M 4 3
Q 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2
X 2
M 2 5
M 3 3
M 2 1
M 4 18
Q 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4
X 3
M 2 6
M 4 1
M 1 1
M 4 15
Q 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2
X 1
M 2 5
M 1 9
M 3 1
M 1 2
M 4 15
M 2 1
M 4 1
Q 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4
X 3
M 1 6
M 4 5
M 1 9
M 4 1
Q 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2
X 1
M 1 2
M 3 1
M 1 1
M 3 1
M 1 1
M 4 9
Q 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4
X 1
M 2 2
M 3 1
M 2 1
M 4 17
Q 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2
X 1
M 3 5
M 2 6
M 4 6

M 2 6
M 4 2
Q 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1
X 1
M 1 6
M 4 18
Q 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1
X 2
M 1 6
M 4 17
Q 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1
X 3
M 2 2
M 3 1
M 2 1
M 4 17
Q 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1
X 1
M 2 6
M 4 12
M 3 1
M 4 4
M 1 1
M 4 1
Q 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3
X 2
M 2 4
M 1 3
M 2 1
M 1 1
M 4 6
Q 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2
X 2
M 3 2
M 1 2
M 2 8
M 4 11
M 3 1
M 4 1
M 2 4
M 4 5
Q 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1
X 3
M 2 5
M 3 1
M 2 1
M 4 14
M 1 1
M 4 1
Q 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2
X 3
M 2 3
M 4 15
Q 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1
X 1
M 1 1
M 4 2
M 1 5
M 4 3

```
M 1 6
M 4 2
Q 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3
X 1
M 2 3
M 4 13
Q 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3
X 3
M 1 1
M 4 7
Q 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2
X 2
M 2 5
M 4 9
M 2 1
M 4 2
M 2 1
M 4 1
Q 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4
X 2
M 2 1
M 3 3
M 1 5
M 3 1
M 4 6
M 1 5
M 3 3
M 2 1
M 4 3
M 1 1
M 4 1
Q 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1
X 2
M 3 1
M 1 5
M 4 14
Q 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3
X 1
M 1 5
M 4 5
M 1 8
M 4 1
Q 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4
X 3
M 2 5
M 4 8
M 1 2
M 4 2
Q 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4
X 3
M 1 2
M 4 5
Q 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4
X 2
M 3 1
M 2 1
M 4 1
M 1 4
M 3 1
```

M 1 1
M 4 8
Q 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2
X 3
M 3 1
M 1 6
M 4 14
Q 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3
X 1
M 3 1
M 2 3
M 4 6
Q 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4
X 1
M 2 8
M 1 1
M 4 11
Q 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1
X 2
M 1 7
M 2 1
M 4 12
Q 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1
X 2
M 3 1
M 4 4
Q 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1
X 3
M 3 1
M 2 5
M 4 13
Q 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4
X 3
M 2 6
M 4 11
Q 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1
X 2
M 2 5
M 3 1
M 2 1
M 4 7
M 1 1
M 4 2
Q 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4
X 3
M 2 5
M 3 1
M 4 2
M 2 1
M 4 3
M 2 3
M 3 2
M 2 1
M 3 1
M 1 1
M 4 1
M 3 1
M 4 4
Q 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2

X 2
M 1 5
M 4 11
Q 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3
X 2
M 1 6
M 4 10
Q 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4
X 2
M 3 1
M 2 6
M 4 1
M 2 1
M 4 10
Q 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1
X 2
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 4
M 2 4
M 4 1
M 3 1
M 4 3
Q 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1
X 3
M 1 3
M 4 10
Q 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1
X 3
M 2 5
M 4 5
M 2 7
M 1 1
M 4 4
Q 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3
X 1
M 2 5
M 4 5
M 2 5
M 4 4
Q 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1
X 3
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 3
M 2 1
M 4 1
M 2 9
M 4 3
Q 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2
X 1

M 2 5
M 4 5
M 2 10
M 4 2
Q 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1
X 1
M 1 3
M 3 1
M 1 1
M 4 1
M 1 1
M 4 2
M 2 1
M 4 9
M 1 2
M 4 1
M 1 5
M 4 2
M 1 2
M 4 1
Q 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2
X 1
M 1 4
M 4 12
M 2 1
M 4 4
Q 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1
X 2
M 3 1
M 2 5
M 4 4
M 2 1
M 4 4
M 1 1
M 4 1
Q 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4
X 2
M 1 3
M 4 15
Q 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1
X 1
M 1 2
M 4 9
M 2 1
M 4 1
M 2 1
M 4 4
Q 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3
X 1
M 1 3
M 3 1
M 2 10
M 4 9
Q 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4
X 2
M 3 1
M 1 3
M 4 7
M 1 1

M 4 7
Q 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1
X 1
M 1 3
M 3 1
M 2 8
M 4 3
M 3 1
M 4 2
M 2 5
M 4 1
M 3 1
M 4 4
Q 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4
X 1
M 3 3
M 1 3
M 4 9
M 2 1
M 4 1
M 2 1
M 4 4
Q 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3
X 1
M 2 3
M 3 3
M 2 1
M 4 5
M 2 2
M 4 1
M 2 5
M 4 1
M 3 4
M 4 2
Q 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2
X 2
M 3 2
M 2 2
M 4 2
M 2 4
M 4 3
M 2 9
M 4 2
Q 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3
X 3
M 3 1
M 1 2
M 3 3
M 1 1
M 4 12
Q 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1
X 1
M 1 3
M 4 7
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1

M 4 1
M 1 1
M 4 2
Q 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2
X 1
M 1 3
M 4 3
M 1 2
M 4 2
M 1 4
M 4 2
M 1 1
M 4 2
Q 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3
X 2
M 1 3
M 4 7
M 2 1
M 4 4
Q 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3
X 3
M 3 3
M 1 1
M 3 3
M 2 3
M 4 6
Q 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1
X 3
M 2 4
M 3 1
M 1 19
M 4 3
Q 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1
X 2
M 3 3
M 1 1
M 3 2
M 1 3
M 4 11
Q 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1
X 3
M 1 2
M 4 8
M 2 3
M 4 2
M 2 1
M 4 2
Q 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2
X 1
M 1 4
M 4 6
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 2
M 4 1

M 2 1
M 4 1
Q 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3
X 1
M 1 5
M 4 13
Q 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2
X 2
M 3 1
M 1 5
M 4 9
Q 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1
X 1
M 1 2
M 4 8
M 2 2
M 4 2
M 2 1
M 4 1
Q 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2
X 1
M 1 6
M 4 8
Q 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2
X 2
M 4 1
M 2 5
M 4 4
M 2 5
M 4 2
Q 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4
X 3
M 2 1
M 1 3
M 2 5
M 1 3
M 3 2
M 1 9
M 4 2
M 1 1
M 4 1
M 1 1
M 4 1
Q 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1
X 2
M 1 2
M 3 1
M 1 2
M 4 7
Q 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4
X 1
M 1 1
M 2 4
M 4 4
Q 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2
X 3
M 1 2
M 4 7
M 2 2

M 4 4
Q 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4
X 1
M 3 5
M 1 2
M 3 1
M 4 8
M 2 1
M 3 1
M 4 3
Q 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4
X 2
M 1 2
M 2 6
M 4 1
M 3 1
M 2 1
M 4 4
M 2 1
M 4 4
Q 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3
X 3
M 1 2
M 4 8
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
Q 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4
X 1
M 2 3
M 4 3
M 2 1
M 4 5
Q 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2
X 1
M 3 1
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 4 5
Q 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2
X 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 4
Q 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1
X 2
M 4 6
Q 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3
X 1
M 1 3
M 3 3

```
M 2 1
M 4 8
M 3 3
M 2 1
M 4 1
M 3 8
M 2 1
M 3 1
M 1 1
M 3 2
M 4 1
M 3 2
M 1 1
M 4 1
Q 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2
X 2
M 1 3
M 4 6
M 1 1
M 4 4
Q 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4
X 3
M 1 3
M 4 10
Q 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2
X 2
M 2 8
M 1 1
M 4 8
Q 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4
X 1
M 1 2
M 3 1
M 2 18
M 4 4
Q 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3
X 2
M 3 3
M 1 3
M 4 16
Q 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3
X 3
M 4 1
M 1 4
M 3 1
M 4 14
Q 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4
X 2
M 1 3
M 4 10
M 2 1
M 4 1
M 2 3
M 4 1
Q 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4
X 2
M 1 4
M 4 12
M 2 2
```

```
M 4 2
Q 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3
X 2
M 1 5
M 4 14
Q 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2
X 2
M 3 1
M 1 3
M 4 10
M 2 2
M 4 1
M 2 1
M 4 2
Q 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3
X 1
M 1 3
M 4 10
M 1 3
M 4 2
M 1 3
M 3 1
M 4 5
M 1 2
M 4 1
Q 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4
X 1
M 3 2
M 1 3
M 4 5
M 1 2
M 3 4
M 4 6
M 1 1
M 4 1
M 1 6
M 3 1
M 4 2
Q 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2
X 1
M 3 1
M 1 2
M 4 3
M 1 1
M 4 10
Q 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3
X 3
M 1 5
M 4 8
M 1 1
M 4 5
Q 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3
X 3
M 3 1
M 1 5
M 4 6
M 2 2
M 4 1
M 2 1
```

```
M 4 2
M 2 1
M 4 3
Q 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3
X 2
M 1 1
M 3 2
M 1 1
M 3 1
M 1 4
M 4 11
Q 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1
X 3
M 3 1
M 2 3
M 4 1
M 2 1
M 4 5
Q 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1
X 1
M 1 2
M 3 1
M 1 1
M 4 12
Q 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2
X 2
M 1 1
M 3 2
M 1 2
M 4 10
M 2 2
M 4 1
Q 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4
X 1
M 3 4
M 1 3
M 4 3
M 1 1
M 4 7
Q 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2
X 2
M 3 2
M 1 8
M 4 4
Q 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4
X 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 4
Q 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1
X 1
M 2 1
M 4 1
M 3 1
M 1 11
M 4 2
Q 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4
```

X 2
M 1 3
M 4 9
Q 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2
X 3
M 3 1
M 1 3
M 4 1
M 1 1
M 4 1
M 1 1
M 4 7
Q 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1
X 1
M 1 3
M 4 6
M 1 1
M 4 2
Q 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1
X 1
M 3 5
M 4 1
M 1 1
M 4 6
Q 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4
X 3
M 3 3
M 2 6
M 4 1
M 2 1
M 4 7
Q 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3
X 3
M 1 6
M 4 8
Q 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3
X 1
M 1 2
M 4 9
Q 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4
X 2
M 3 1
M 4 4
M 2 4
M 4 1
M 2 13
M 4 1
Q 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2
X 1
M 1 1
M 3 1
M 1 4
M 4 8
Q 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1
X 1
M 2 7
M 4 1
M 2 6
M 4 2

Q 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1
X 3
M 1 3
M 4 8
Q 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2
X 1
M 1 2
M 4 8
Q 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4
X 1
M 3 3
M 2 5
M 4 3
M 2 1
M 4 1
M 1 1
M 4 4
Q 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2
X 2
M 3 2
M 2 9
M 4 2
M 2 1
M 4 2
Q 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2
X 3
M 2 7
M 4 10
Q 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3
X 3
M 3 1
M 1 6
M 4 10
Q 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3
X 1
M 4 6
M 1 1
M 4 4
Q 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3
X 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 7
Q 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1
X 1
M 2 6
M 4 1
M 2 1
M 4 15
Q 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4
X 2
M 3 1
M 4 7
M 2 1
M 4 6
Q 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4

X 1
M 2 7
M 4 17
Q 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1
X 1
M 4 1
M 3 1
M 1 3
M 4 6
M 2 1
M 4 6
Q 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1
X 1
M 2 7
M 4 15
Q 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3
X 3
M 1 3
M 4 6
M 1 1
M 4 10
Q 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4
X 1
M 4 1
M 2 3
M 4 10
M 2 2
M 4 1
M 2 7
M 4 3
Q 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3
X 3
M 3 2
M 2 1
M 4 15
Q 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3
X 3
M 3 2
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 14
Q 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3
X 3
M 4 7
M 1 4
M 4 5
M 1 11
M 4 1
Q 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1
X 1
M 4 7
M 1 4
M 4 5
M 1 1

```
M 4 1
M 1 4
M 3 1
M 4 4
Q 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2
X 3
M 4 8
M 2 5
M 4 4
M 2 7
M 4 3
Q 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4
X 1
M 3 1
M 4 3
M 1 4
M 4 1
M 1 2
M 4 1
M 1 3
M 4 1
M 1 1
M 4 1
Q 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2
X 2
M 4 7
M 2 3
M 4 3
M 2 2
M 4 2
M 3 2
M 2 1
M 4 1
M 2 4
M 3 1
M 4 3
Q 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2
X 3
M 1 8
M 4 2
Q 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2
X 2
M 2 1
M 3 1
M 1 3
M 3 1
M 4 7
M 1 3
M 4 5
M 1 2
M 4 1
M 1 3
M 3 5
M 1 1
M 4 2
M 3 1
M 4 1
Q 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2
X 1
```

M 4 8
M 1 5
M 4 4
M 1 6
M 4 1
Q 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1
X 3
M 4 9
M 1 3
M 4 2
M 1 1
M 4 1
M 1 13
M 4 1
Q 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1
X 3
M 3 1
M 4 7
M 3 3
M 2 3
M 4 2
M 2 1
M 4 1
M 2 1
M 4 1
M 1 1
M 4 6
Q 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3
X 2
M 2 2
M 3 1
M 4 7
M 3 1
M 4 2
M 2 2
M 4 1
M 2 1
M 4 1
M 1 5
M 4 6
Q 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2
X 3
M 3 1
M 1 2
M 4 6
M 1 2
M 4 6
M 1 7
M 4 2
Q 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2
X 2
M 3 3
M 2 1
M 4 11
M 2 1
M 4 1
M 2 1
M 4 2
M 1 1

M 4 1
M 2 1
M 3 1
M 2 5
M 3 3
M 2 1
M 4 2
Q 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3
X 3
M 3 1
M 4 9
M 1 2
M 4 7
Q 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2
X 1
M 3 3
M 1 5
M 4 17
Q 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4
X 1
M 4 4
M 1 6
M 4 1
M 1 2
M 4 1
Q 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1
X 3
M 2 2
M 4 5
M 2 3
M 3 1
M 4 6
M 2 1
M 4 2
M 2 6
M 3 1
M 2 5
M 4 1
Q 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4
X 3
M 1 3
M 4 11
M 1 3
M 4 1
M 1 3
M 4 2
Q 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3
X 1
M 4 7
M 1 3
M 4 2
M 1 1
M 4 1
M 1 1
M 4 2
M 1 10
M 4 1
Q 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2
X 3

```
M 2 1
M 4 8
M 1 4
M 4 1
M 1 1
M 4 7
Q 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1
X 3
M 2 2
M 4 8
M 2 1
M 4 1
M 1 1
M 4 7
M 1 1
M 4 1
Q 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4
X 3
M 2 2
M 4 7
M 2 1
M 4 4
M 2 1
M 4 1
M 2 13
M 4 1
Q 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3
X 2
M 3 3
M 4 11
M 2 5
M 4 2
M 2 8
M 4 1
M 2 1
M 4 1
Q 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4
X 1
M 4 5
M 2 3
M 4 7
M 2 7
M 4 1
M 3 1
M 4 1
Q 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3
X 2
M 2 3
M 3 1
M 4 17
Q 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3
X 2
M 2 3
M 4 8
M 2 1
M 4 4
Q 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4
X 2
M 2 6
```

```
M 4 6
M 2 3
M 4 4
M 2 1
M 4 3
M 2 1
M 4 3
Q 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4
X 3
M 4 3
M 3 1
M 4 5
M 1 1
M 4 1
M 1 1
M 4 7
Q 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3
X 3
M 2 3
M 4 7
M 2 1
M 4 1
M 2 1
M 4 7
Q 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3
X 2
M 3 1
M 4 11
M 2 1
M 4 1
M 1 1
M 4 3
Q 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2
X 1
M 2 1
M 4 9
M 2 1
M 3 2
M 4 5
Q 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2
X 1
M 4 7
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 3
M 4 1
Q 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1
X 2
M 3 3
M 2 3
M 4 14
```

Q 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4
X 1
M 1 6
M 4 15
Q 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1
X 1
M 3 2
M 1 2
M 4 11
M 1 1
M 4 5
Q 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4
X 1
M 3 1
M 2 6
M 4 7
M 2 1
M 4 8
Q 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2
X 2
M 3 3
M 1 2
M 3 3
M 1 2
M 4 11
M 1 1
M 4 5
Q 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2
X 1
M 4 6
M 1 1
M 3 1
M 1 4
M 4 10
Q 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1
X 2
M 4 8
M 2 6
M 4 4
M 2 5
M 4 4
Q 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4
X 1
M 4 6
M 1 6
M 2 9
M 4 3
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 5
M 4 3
Q 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2
X 2
M 1 5
M 4 5

```
M 1 4
M 4 1
Q 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3
X 3
M 2 1
M 3 1
M 4 7
M 2 3
M 4 4
M 2 1
M 4 1
M 2 8
M 4 1
Q 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4
X 2
M 3 1
M 4 7
M 1 2
M 4 2
M 3 1
M 2 6
M 4 1
M 2 1
M 4 7
Q 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2
X 2
M 3 3
M 1 1
M 4 16
Q 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1
X 1
M 3 2
M 2 5
M 4 11
M 2 1
M 4 5
Q 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4
X 3
M 4 11
M 2 1
M 4 5
Q 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4
X 2
M 4 8
M 2 2
M 4 1
M 2 1
M 4 7
Q 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1
X 3
M 3 3
M 1 5
M 4 5
M 1 2
M 4 1
M 1 3
M 4 5
Q 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2
X 2
```

M 4 8
M 2 6
M 4 4
M 2 1
M 4 1
M 2 3
M 4 1
Q 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3
X 3
M 4 6
M 3 1
M 4 3
M 2 6
M 4 3
M 2 1
M 4 1
M 2 3
M 3 1
M 4 1
M 2 1
M 4 3
Q 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1
X 2
M 3 1
M 4 7
M 1 5
M 4 5
M 1 7
M 4 1
Q 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1
X 2
M 3 1
M 4 17
Q 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1
X 2
M 3 1
M 1 2
M 4 16
Q 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3
X 3
M 4 1
M 3 3
M 1 4
M 4 12
M 1 1
M 4 4
Q 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4
X 2
M 4 7
M 1 3
M 4 1
M 1 1
M 4 3
M 1 1
M 4 6
Q 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4
X 2
M 3 3
M 4 8

M 1 2
M 4 8
Q 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4
X 1
M 2 6
M 4 5
M 2 8
M 4 2
Q 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3
X 2
M 4 7
M 2 6
M 4 10
Q 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4
X 2
M 3 1
M 4 3
M 2 5
M 4 1
M 2 1
M 4 1
M 2 6
M 4 1
Q 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2
X 3
M 1 5
M 4 5
M 1 3
M 4 1
M 3 1
M 1 1
M 4 3
Q 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2
X 3
M 3 2
M 4 12
M 1 1
M 4 3
Q 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2
X 1
M 4 4
M 1 2
M 4 1
M 1 14
M 4 1
Q 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4
X 1
M 4 4
M 1 5
M 4 1
M 1 7
M 4 1
Q 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4 4 1
X 1
M 3 1
M 4 8
M 2 8
M 4 4
M 2 2

```
M 4 1
M 2 1
M 4 1
0
Q 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4 4 1 6 1
X 1
```

Lássuk a visszajátszást. Itt mindenekelőtt fölmerül a kérdés: honnan fogja tudni a program, hogy tetrisezünk vagy filmet játszunk vissza? Maradjunk abban, hogy egyelőre sehonnan, mármint egyelőre nem lesz semmilyen menü, amiből a játékos kiválaszthatná, hogy mit akar csinálni: beleírjuk a programba fixen, ott kell átírni. Menü később lesz.

A főprogram összesen három végrehajtható utasításból áll: **inic**, **aknarajz** és **mozgas** hívásából. **Inic**re mindenképpen szükség van, hiszen ő tölti föl falakkal az akna széleit, ő ad értéket a szükséges változóknak, ő definiálja a palettát, ezek mind-mind kellenek a visszajátszásnál is. Viszont a másik két rutinhívás csak akkor kell, ha tetrisezünk. Betehetjük tehát őket egy elágazásba:

SELECT CASE 1
CASE 0: aknarajz: mozgas
END SELECT

E pillanattól tehát ezek nem hajtódnak végre, nem lehet tetrisezni a programmal, csak ha a **SELECT CASE** feltételét átírjuk 0-ra. A **CASE 1** ág lesz a visszajátszás. Ez indítson el egy külön visszajátszó (angolul *replay*) rutint, ami kapja meg a filmfile nevét.

CASE 1: replay "tetris.flm"

Eddig rendben, lássuk magát a visszajátszást.

Az persze egy pillanatig sem kétséges, hogy a korábban megírt programrészek közül föl kell használnunk mindent, amit csak lehet. A kérdés az, hogy mit lehet fölhasználni.

Kezdjük az elején: a pálya beolvasásával. A **palya** rutin kicsit problematikus, mert konkrétan meg van neki mondva, hogy milyen file-ból olvasson, és pályaszámot is keres, ami a filmfile-unkban nincsen. De ezt is viszonylag egyszerűen meg lehet oldani. Mondjuk ki, hogy ha **ply**-ben nulla van, akkor semmi keresgélés nem történik, egyből olvasni kezdi a file-t ott, ahol éppen tart. Nem is nyitja meg, mert már nyitva lesz. És persze a **film** rutint se hívogatja. Tehát a rutin elején levő két **OPEN** utasítást **IF ply THEN** mögé tesszük. A **prg** tömböt nullázó sor és a két következő marad, az **IF**-nek viszont azt kell mondani, hogy ha **ply** nulla, akkor minden körülmények között kezdje meg a feldolgozást. Ez egyszerűen úgy fog menni, hogy a feltételhez hozzábiggyesztjük: **OR ply = 0**. A három sorral lejjebb álló **PRINT** a filmkészítéshez tartozik, **IF ply THEN** kell elé, akárcsak két későbbi társánál. De ugyanez a teendő a rutin végén levő **CLOSE** és **film** utasításokkal is, ezeket is csak akkor hajtjuk végre, amikor tetrisezünk.

Eddig rendben, a **palya** rutint hozzáidomítottuk az új szituációhoz; hívjuk meg. Kell egy **replay** rutin, aminek paramétere a lejátszandó film neve. Először is nyissa meg ezt a file-t, aztán töltse be belőle a pályát és rajzolja ki.

```
SUB replay (f$)
OPEN f$ FOR INPUT AS 2
palya 0
aknarajz
```

A pályát betöltöttük, dolgozzuk föl a film további részét. Kell egy ciklus, ami beolvas egy sort a filmből, feldolgozza, megint olvas, feldolgoz.

```
DO UNTIL EOF(2)
LINE INPUT #2, l$
SELECT CASE word$(l$)
```

Kézenfekvő ötlet a sorok első szavait venni a **word$** segítségével, hiszen úgy írtuk meg a file-t, hogy a sorok első (egybetűs) szavai jelzik, mi történik. Milyen szavaink is voltak? A file-ban konkrétan egy `Q` következik a pályarajz után, ez a **queue**-t tölti fel tartalommal. Megtehetnénk, hogy átírjuk **ujelem**et is, ahogy a **palya** esetében tettük, de ez fölösleges macera lenne, sokkal egyszerűbben megoldható a dolog:

```
CASE "Q": queue = ""
DO UNTIL l$ = "": queue = queue + CHR$(VAL(word$(l$))): LOOP
```

Ezért fölösleges lett volna bíbelődni. Viszont van egy gond: a **queue**-t **ujelem** rajzolja ki a képernyő sarkába, nem más. Mondjunk le a kirajzolásról visszajátszás közben? Ugyan. Tegyük **ujelem**nek ezt a részét külön rutinba, és hívjuk meg onnan is, innen is. Legyen a rutin neve **queuerajz**, paramétere nem lesz.

Nézzük tovább. A film következő parancsa az `X`, ami **ext** értékét állítja be. Megvalósítása kézenfekvő:

```
CASE "X": ext = VAL(word$(l$))
```

Következik az `M` parancs, az elem mozgatása; többféle parancs nincs is a file-ban. De ennél már van morfondíroznivaló.

Nagyon jó lenne, ha az `M` parancsot a **mozgas** rutin hajtaná végre, mert az gondoskodna minden szükséges dolog kirajzolásáról, adminisztrálásáról, egyszóval mindenről. De hogyan csináljuk?

A **mozgas** a programunk magja, ez hívja ismétlődően a program többi részét. Ezért van benne egy nagy ciklus, ami a rutin egy híján az összes utasítását tartalmazza. Azt nem tudjuk megcsinálni, hogy **IF akármi THEN DO**, aztán a rutin végén **IF akármi THEN LOOP**, mert a QB hibaüzenetet ad, egy ciklust be lehet tenni egy feltételes utasításba, de a ciklus fejét vagy végét külön-külön nem. Viszont azt meg lehet tenni, hogy a ciklus végére odaírjuk ezt az akármit feltételként; ő egy globális változó lesz mondjuk **uzemmod** néven, az értéke 0, ha tetrisezünk és 1, ha visszajátszunk. Tehát **LOOP UNTIL uzemmod**.

A rutin elején **ujelem** hívása csak akkor kell, ha tetrisezünk, tehát **IF uzemmod = 0 THEN**. **Film** hívásait is ki kellene iktatni, de rögtön az első már egy **IF-**

ben van; ahelyett, hogy az egymásba ágyazott **IF**-ekkel bíbelődnénk, egyszerűen tegyük be **film**be, hogy ha **uzemmod = 1**, akkor lépjen ki.

Mozgasban már csak egy helyen kell változtatnunk: lejátszáskor nem a billentyűzetről olvasunk, hanem a file-ban talált számok alapján mozgatjuk az elemet. Tehát az **a$ = INKEY$** és az azt követő **IF** bekerül egy **SELECT CASE uzemmod** utasítás **CASE 0** ágába, a **CASE 1** ágba pedig betesszük a filmben talált számok szerinti visszajátszást.

Ezeket a számokat **replay** át kell, hogy adja **mozgas**nak. Erre remekül megfelel a **billn** és **billk** változó, hiszen azok korábban is ezt a funkciót látták el, csak akkor a billentyűlenyomásokat rögzítették és tették file-ba kiírhatóvá, most pedig éppen fordítva, a file-ból érkező adatokat alakítják „billentyűlenyomásokká". Tehát **replay CASE "M"** ága így szól:

CASE "M": billk = VAL(word$(l$)): billn = VAL(word$(l$)): mozgas

A **mozgas**ban most létrehozott elágazás **CASE 1** ága pedig ezzel kezdődik majd:

a$ = CHR$(0) + MID$("KMHP", billk, 1)

Billk-t itt visszaalakítottuk ugyanolyan kóddá, amit az **INKEY$** is ad. Mi legyen **billn**-nel? Két választásunk is van: **mozgas** addig csökkentheti az értékét, amíg nulla nem lesz, minden alkalommal végrehajtva egy mozdítást, avagy csinálhatja ugyanezt **replay**. Egy biztos: gondoskodni kell arról, hogy **billk** értéke ne változhasson ezenközben. A változó a jelenlegi elágazásunk **CASE 0** ágában is értéket kap, de az most nem fog végrehajtódni; szerepel még a rutin végén, a lefelé mozdításnál is, ahol azonban ki kell iktatni. Ez a sor tehát **IF uzemmod = 0 THEN** mögé kerül.

Tegyük **replay**-be **mozgas** ismételt hívogatását. A **CASE "M"** ágába ekkor ez kerül:

DO: mozgas: billn = billn - 1: LOOP UNTIL billn = 0

Elrendeztünk mindent? Elméletileg igen. Nincs más dolgunk, mint kipróbálni a programot.

Két probléma akad: az akna legalsó sorában kapcsolók láthatók az oda való tartalom helyett, és a program az első elem megjelenése után billentyűlenyomásra vár. És szépen mozgatja a billentyűk nyomkodása szerint. Persze: **uzemmod** beállításáról megfeledkeztünk. Ez történjen a **replay** rutin elején. A kapcsolókat egyelőre hagyjuk.

Hibaüzenetet kapunk **elemrajz**ból: **Subscript out of range**. **E** és **i** értéke nulla, márpedig a **raktar** tömbben ilyen indexek nincsenek. Persze, **mozgas** semmit se tud arról, hogy milyen elemet kellene éppen mozgatnia. Sőt arról sem, hogy az akna mely pontján. Nincs más választásunk, ezt a négy változót oda kell adnunk **mozgas**nak. Legyen belőlük paraméter.

Emiatt át kell írnunk a rutin hívását a főprogramban; itt tegyünk egyszerűen négy 0-t. És át kell írnunk a hívását **replay**-ben is, itt viszont **e, i, x, y** legyen a paraméterlista, és gondoskodnunk kell az értékek karbantartásáról.

E és **i** értéket kaphat a "Q" parancsnál, mert amikor új elemet sorsoltunk, mindig kiírtunk a filmbe egy ilyen parancsot. Ide kerülhet tehát **e = ASC(queue): i = ASC(MID$(queue, 2))**. Ami a koordinátákat illeti, ugyanitt állítsuk be őket ugyanúgy, ahogyan **ujelem** teszi: **x = aknax \ 2: y = 2**. Elméletileg másra nem lesz szükség, mert **mozgas** már önmagától karbantartja a koordinátákat. Hiszen megváltoztatja **x** és **y** értékeit, azok mint paraméterek visszakerülnek **replay**-be, és a következő hívásnál **mozgas** ezeket az értékeket kapja paraméterül.

A program indítása váratlan eredményt szolgáltat. Az elemek eszeveszett vibrálásba kezdenek a képernyő tetején, ahol – a bal oldali falon – kockahalom alakul ki. Az akna alján levő kapcsolók alighanem följebb tolták eggyel az akna tartalmát, ezért a legelső elem elakadt a falban; onnantól aztán nem csoda, hogy hibás eredményt kapunk. Nézzünk utána azoknak a kapcsolóknak.

Nyilván azért vannak itt kapcsolók, mert a pályarajz után következő sort (az első kapcsolóhoz tartozó Kate nyelvű programot) is beolvasta **palya**. Gyors ellenőrzéssel kiderül, hogy a pályarajz a filmfile-ban is 19 sorból áll, tehát csak úgy kerülhettek följebb a sorok, hogy valaki beolvasta az első sort, mielőtt **palya** munkához láthatott volna. Ez a valaki pedig csak maga **palya** lehetett; bizonyára beolvasta az első sort, úgy döntött, hogy az nem jó neki, de a másodiknál már elkezdte dekódolni a pályarajzot. De ha figyelmesen újraolvassuk a rutin szövegét, kiderül, hogy nem ez a helyzet. A 9. sorbeli **LINE INPUT** által beolvasott sort az **IF** szabályszerűen kiértékeli és igaz eredményre jut (az **OR ply = 0** jóvoltából), de aztán a 12. sor beolvas egy új sort a pályarajzból, mielőtt bármit is tehetnénk. Csakugyan: a 9. sor a pályaszámot tartalmazó sor beolvasására lett kitalálva. Tegyük tehát **IF uzemmod = 0 THEN** mögé.

Az eredmény annyiban változott, hogy az akna most már helyes képet mutat, nincsenek kapcsolók az alján, viszont továbbra is kockahalom alakul ki fent. Az események túl gyorsak ahhoz, hogy követni tudjuk őket, tegyünk hát egy **a$ = INPUT$(1)** utasítást **replay**-be, közvetlenül **mozgas** hívása után.

A dolog nem a legsikerültebb: vagy egy tucat billentyűlenyomás után jelenik meg egy fekvő **I** elem a bal oldali fal tetején, s nem derül ki, hogy mi volt az útvonala. A további elemekkel is hasonló a helyzet. Próbáljuk áttenni az utasítást **mozgas**ba, a **CASE 1** ágba. Ott már van egy értékadás **a$**-nak, eléje tegyük.

Most már látszik, hogy mit csinálnak az elemek. Zagyvaságot. Egy függőleges **I** elem négyet mozdul balra, aztán vízszintesbe fordul, ötöt megy lefelé és kiköt a fal tetején. A későbbi elemek még rosszabbak, többségük egy darabig szaladgál, aztán eltűnik valahol a semmiben, az akna közepén.

Debugoljunk. Írassuk ki **mozgas**sal, hogy éppen milyen paraméterek alapján cselekszik. **LOCATE 1, 2: PRINT billk; billn**.

Ezek az adatok jelennek meg: **1 4** (az elem négyet balra lép, közben a második szám 1-re csökken); **3 1** (az elem elfordul vízszintesbe); **4 13** (az elem elindul lefelé, de amikor a második szám 9-re csökken, beleütközik a falba, ezért új **I** elem jelenik meg az akna közepén, függőlegesen – a második szám tovább számlál

visszafelé); **2 3** (az elem elindul jobbra); **4 1** (az elem lejjebb megy); **2 1** (jobbra megy); **4 1** (lejjebb megy); **2 1** (nem történik semmi, nem tud már jobbra menni); **1 3** (hármat balra megy); **4 3** (hármat ereszkedik); **2 5** (az elem eltűnik az akna közepén és egy fekvő **H** elem jelenik meg fönt)…

Nézzünk utána ezeknek a számpároknak a filmfile-ban. Megtaláljuk őket, `M` utasításokban követik egymást, ebben a sorrendben, csakhogy nem az első, hanem a *harmadik* pályarajz utáni első elemhez tartoznak. Kellemetlen. Valahol a programunk teljesen átugrott két pályát.

Mivel **replay** dolgait ismerjük, lehet, hogy **palya** lesz a bűnös. Két helyen olvas a file-ból, a 12. és a 27. sorban. A 12. aligha felelős, mert egy ciklusban van, ami világosan megmondja, hogy csak 19-szer hajtódhat végre. Nézzük a másikat. Tegyük mögé, hogy **PRINT I$**.

A program szépen kilistázza a filmfile szövegét a grafikus képernyő hatalmas betűivel, öröm nézni. Itt van tehát a baj.

Tegyünk **STOP**-ot ennek a **PRINT**-nek a helyére, és nézzük meg *F8*-cal, merre megy innen a végrehajtás.

Érdekes. Egy darabig szaladgál ez a ciklus, aztán egyszer csak végre akar hajtódni a 39. sorbeli **CLOSE 2, 3: film 0** rész, ami elé oda van írva, hogy **IF ply THEN**. És **ply** értéke valami okból 1. Mégis mitől?

A *Calls* menü adja meg a választ: ez visszavisz oda, ahonnan a **palya** rutint hívták. Ez pedig **inic**! Itt van egy **ply = 1** és egy **palya ply** utasítás. Tegyük őket **IF uzemmod = 0 THEN** mögé.

Az eredmény a **STOP** kiiktatása után továbbra is ugyanaz, az elemek pontosan ugyanazokat a zagyva pályákat írják le. Tegyük vissza a **STOP**-ot és nézzük meg újra ezt a részt. Még mindig **inic** hajtja végre.

Ó, persze. Hiszen **uzemmod** értékét csak a **replay** rutin állítja 1-re, ami **inic** után hajtódik végre. **Inic** a program legelején áll, itt még nulla van a változóban. Tegyük át az értékadást **inic** elé a főprogramba.

Még mindig nincs változás. Viszont most már nem **inic** hívja a **palya** rutint, hanem **replay**. Kövessük a **STOP**-tól a dolgokat *F8*-cal. Ha a **LINE INPUT** után kiírogatjuk **I$** értékét, kiderül, hogy ez a ciklus nem áll meg, amikor túljutottunk a változókat tartalmazó sorokon és a lezáró nullán, hanem tovább olvas. Ez nem szép tőle.

Tanulmányozzuk a ciklus kilépési feltételét. A 34. sorban **IF VAL(I$) THEN EXIT DO** áll. Ez benne van egy másik **IF**-ben, pontosabban annak **ELSE** ágában, de ez most nem lényeges; a pályaleírást egy nullával zártuk le, márpedig **VAL(I$)** értéke ez esetben nulla lesz. Nincs kilépés. Jobb lesz, ha a **LINE INPUT** után beteszünk egy külön kilépési feltételt, ami speciálisan erre az esetre vonatkozik:

IF ply = 0 AND I$ = "0" THEN EXIT DO

A helyzet változott. Az elemek más pályát futnak be, de ez a pálya még mindig zagyva. Még mindig eltűnnek egyes elemek a levegőben, más elemek nem oda mennek, ahova kell nekik, egyáltalán nem jó az egész.

Kezdjük elölről a debugolást a korábbi módszerrel. Figyeljük a számokat.

4 8 (fekvő **I** elem látható, ereszkedni kezd); **2 5** (kettőt jobbra megy, aztán nekiütközik a jobb oldali falnak, a további számlálásnál már nem tud mozdulni); **4 10** (leereszkedik az akna fenekére). Keressük meg ezeket a számokat a file-ban.

Nagyon hamar megtaláljuk: ezek a legelső elem mozgását írják le. Most már tehát jól „céloztunk" a file-ban, de vagy rossz számok vannak ott, vagy rosszul értelmezzük azokat. Ezt a legkönnyebben úgy tudjuk meg, hogy átállítjuk a programot játékra és csinálunk egy új felvételt. Elég egyetlen elem.

Az átállítás úgy a legkönnyebb, hogy **uzemmod**ot 0-ra állítjuk, és az 1-es szám helyett ezt a változót tesszük a **SELECT CASE**-be.

Egy **V** elemet kaptam, amit tizenkétszer balra vittem, kétszer elfordítottam, aztán levittem a padlóig tizenhét gombnyomással. Ez került a file-ba:

```
Q 4 1 6 4 8 1 8 1 11 1 7 3 12 4 4 4 9 1 2 2
X 2
M 1 12
M 3 2
M 4 17
Q 6 4 8 1 8 1 11 1 7 3 12 4 4 4 9 1 2 2 4 1
X 3
```

Nézzük meg a visszajátszást. Írjuk át a főprogramban a **"tetris.flm"**-et **"ujtetris.flm"**-re.

Az elem pontosan azt a pályát futja be, amit a rögzítéskor csináltam vele, balra megy tizenkétszer, elfordul kétszer és lejön. Csakhogy ez egy másik elem: egy **Z**.

Nem csoda. Világosan ott áll a `Q` parancsban: `4 1`. A **Z** elem *LISZTOJ*-kódja 4, és az első állásban jelent meg.

Ez probléma. Ha a visszajátszással lenne baj, azt kijavíthatnánk, de ha a rögzítésnél van a hiba, akkor a **tetris.flm** fabatkát sem ér, eldobhatjuk.

De hátha mégsem. A játék első elemét a program tökéletesen kifelejtette, de a későbbiek megvannak, mert azokat már mind a **queue**-ból veszi, és azt viszont rögzítette. Remélhetőleg jól. Ha kiderítjük, mi volt az első elem (ki emlékszik már arra?), akkor lehet, hogy a filmünk megint lejátszhatóvá tehető.

Hát derítsük ki, mi volt az első elem. Fogjunk egy szövegszerkesztőt és írjunk be a **tetris.flm** első `Q` parancsának elejére egy új elemet. Mentsük ki a file-t, kezdjük el a lejátszást. Ha nem jó, írjunk másik elemet. Legegyszerűbb, ha végigmegyünk a számokon 1-től kezdve.

Nem kell végigmenni az összes elemen ahhoz, hogy kiderüljön: itt bizony más bajnak is kell lennie. Az első elemet (bármi is az) szépen leteszi a zöld rács jobb oldalára, a következőt viszont úgy hagyja ott baloldalt, hogy nem ér le az akna aljáig, előbb eltűnik. Amikor ennek az elemnek az útját rögzítettük, nyilván azért lehetett előbb abbahagyni a mozgatását, mert ott ért feneket, kockák voltak alatta. De az első elem egész más irányba indul, így nem lesznek alatta kockák. Lehet, hogy a **tetris.flm** mégse lesz menthető. Tegyünk egy új próbát: rögzítsük mondjuk három elem útját, aztán játsszuk vissza.

Az első elem egy **K,** amit a bal oldali fal tetejére tettem. A második **J,** amit úgy fordítottam, hogy három kockája feküdjön, és letettem a zöld rács bal sarkába. A harmadik egy **V,** amit elfordítás nélkül rátettem az előző elem jobb szélső kockájára. A negyedik elem egy bomba, de itt megállítom a rögzítést és átállítom a programot lejátszásra.

A **K** helyett természetesen a **J** elem jelenik meg, ez írja le azt az utat, amit eredetileg a **K** tett meg. Ezután a **V** elem teszi meg a **J** útját, és végül egy bomba (de nem ám 1-es típusú, hanem 2-es) a **V** elem útját.

Írjuk bele a filmfile első `Q` parancsába a **K** elemet. Ez a nyolcadik elem, az állását szerencsére nem kell tudnunk.

A lejátszás váratlan eredményre vezet. A **K** elem megjelenik és leírja ugyanazt a pályát, amit a valóságban is megtett, kiköt a fal tetején. Ezután viszont a **V** elem következik a **J** helyett, ez teszi meg a **J** útját, és utána jön a csőbomba a **V** eredeti útvonalán. A **J** elem teljesen kimarad.

Ez jogos, hiszen a film első két `Q` parancsa így szól:

```
Q 8 1 7 4 11 3 12 2 2 1 1 4 2 3 8 3 2 3 7 1 12 4
Q 11 3 12 2 2 1 1 4 2 3 8 3 2 3 7 1 12 4 11 1
```

A `8 1`-et mi írtuk ide az imént, ez a **K** elemet jelöli. A **J** elem *LISZTOJ*-kódja hetes, de a `7 4` nem jelenik meg a következő `Q` parancsban, hiszen eredetileg az első mondta azt, hogy végy egy hetes kódú elemet a negyedik állásban.

A rögzítésnél tehát egy teljes elemmel elcsúsztunk. A játék legelső elemének kódja elveszett, de az általa megtett lépések nem: a második elem teszi meg azt az utat, mert az ő kódja áll az első elem útvonala előtti `Q` parancsban. Az ő valódi útvonala fölött már a harmadik elem kódja áll, és így tovább.

A programozó a homlokára üt. Hát persze! Amikor a **queue**-t kiírattuk a filmfile-ba, az elem már ott volt az aknában. Amikor egy elem megjelenik az aknában, a **queue**-ból már törlődött, ott a *további* elemek vannak felsorolva, éppen erre való a **queue**. Tehát amikor az első elem megteszi útját az aknában, akkor a **queue** a második elemmel kezdődik, és ez íródik ki a file-ba. Amikor a hatvanadik elem közlekedik az aknában, akkor a **queue** a hatvanegyedik elemmel kezdődik, és az íródik a file-ba.

Ha ez így van, akkor nincs más teendőnk, mint fogni a `7 4`-et és átmásolni a második `Q` parancs elejére, ahonnan viszont a `11 3`-at másoljuk a harmadik `Q` elejére, ahonnan viszont a `12 2`-t másoljuk a negyedik `Q` elejére. Játsszuk le így!

A lejátszás most már tökéletes.

Ha ezt a filmet ki tudtuk javítani, akkor a **tetris.flm** is menthető, csak persze rá kell jönni, hogy mi volt az első elem. Az igaz, hogy ebben a file-ban egy kicsit több `Q` parancs van, szövegszerkesztővel megszámoltattam, 317 darab. De írhatunk egy kis programot a feladatra.

```
DEFINT A-Z
OPEN "tetris.flm" FOR INPUT AS 2
OPEN "tetris.jav" FOR OUTPUT AS 3
```

```
DO UNTIL EOF(2)
LINE INPUT #2, I$
IF LEFT$(I$, 1) = "Q" THEN
q$ = word$(I$)
PRINT #3, "Q " + n1$ + " " + n2$ + " " + I$
n1$ = word$(I$)
n2$ = word$(I$)
ELSE
PRINT #3, I$
END IF
LOOP
CLOSE
```

Ennyi az egész (plusz a **word$** függvény, de azt már ismeri az Olvasó). Nagyon egyszerűen működik. Végigolvassa a file-t elejétől végéig, és a `Q` betűvel kezdődő sorokat keresi. Ha ilyet talál, eldobja (**q$**-ba teszi) magát a betűt, aztán az előző sorból megjegyzett két számot a sor elejére biggyesztve és a `Q` betűt pótolva kiírja a sort; végül pedig kitermeli a sor eredeti szövegéből a két számot a következő sor számára. Persze az első `Q` parancsnál még üres szöveg van **n1$**-ban és **n2$**-ban, de az első elem kódjait úgyis kézzel kell pótolnunk.

A létrejött file remélhetőleg már a helyes kódokat tartalmazza, az első elem kivételével.

Úgy néz ki, hogy igen. Kezdjük újra találgatni az első elemet.

Jó sok számpárt ki kellett próbálni, amíg kiderült, hogy `11 4` kell ide, **V** elem volt a negyedik állásban. Így már végigjátssza a file-t, egészen a zöld rács beteléséig. Akkor leáll **File already open** üzenettel: a file már nyitva van. Ezt a **palya** rutin mondja **tetris.lev** megnyitásánál, no de őneki egyáltalán nem lenne szabad ezt a file-t most megnyitnia. Azért akarja, mert **ply** értéke nem nulla. Persze, mert amikor a rács betelik, automatice növeljük az értékét. Hát lejátszáskor nem szabad. A **racs** rutin erre vonatkozó két (18. és 19.) sorát **IF uzemmod = 0 THEN** mögé kell tennünk.

Sajnos még mindig nem jó: nem jelenik meg a második pálya. Szólnunk kell **replay**-nek, hogy olvassa be és jelenítse meg.

Hogyan? Semmilyen jelet nem tettünk a filmbe, ami az új pályarajz kezdetét jelöli. De talán nem is kell. Ha **racs**nak megengedjük, hogy megnövelje **ply** értékét, de azt megtiltjuk, hogy a **palya** rutint is meghívja, akkor **replay** 1-et fog találni a változóban és tudni fogja, hogy be kell olvasnia az új pályát.

Tehát **racs** 18. sora:

ply = ply + 1: IF uzemmod = 0 THEN palya ply

replay-be pedig ez kerül:

18 IF ply THEN ply = 0: palya 0: aknarajz

A lejátszás most már zavartalanul haladhat tovább a második és a harmadik pályára. Sajnos a harmadik pálya vége felé megint történik valami hiba, amitől a folytatás zagyvasággá változik. Úgy 3300 pontnál tart a játék, amikor ez bekövet-

kezik. Ezzel már ráérünk később foglalkozni. A hárompályás játék legnagyobb része lejátszható.

Tetrisünk veszélyes dologgá változott: most már minden hibát, minden rossz helyre tett elemet látni fog az utókor is!

```
DECLARE SUB mozgas (e%, i%, x%, y%)
DECLARE SUB queuerajz ()
DECLARE SUB replay (f$)
DECLARE SUB film (f%)
DECLARE SUB extra3ko (ko$, x%, y%)
DECLARE SUB extrarajz (x%, y%)
DECLARE FUNCTION filter% (f$, x%, y%)
DECLARE FUNCTION word$ (s$)
DECLARE SUB program (pr$)
DECLARE SUB kapcsolo (be$)
DECLARE SUB exec (cmd$)
DECLARE SUB kapcsrajz (x%, y%, k%)
DECLARE SUB iras (t$, x%, y%, c%)
DECLARE SUB pontozas (p%)
DECLARE SUB palya (ply%)
DECLARE SUB zuditas (y%, k%, v%)
DECLARE SUB kockarajz (x%, y%, szin%, t%)
DECLARE FUNCTION szin% (e%)
DECLARE SUB paletta (p%, r%, g%, B%)
DECLARE SUB racsrajz (x%, y%)
DECLARE SUB kockatorles (x%, y%)
DECLARE SUB elemtorles (e%, i%, ex%, ey%)
DECLARE SUB extra (e%, i%, ex%, ey%)
DECLARE SUB racs ()
DECLARE SUB ujelem (e%, i%, x%, y%)
DECLARE SUB sorok ()
DECLARE SUB ragadas (e%, i%, ex%, ey%)
DECLARE FUNCTION mehet% (e%, i%, x%, y%)
DECLARE SUB elemrajz (e%, i%, x%, y%)
DECLARE SUB inic ()
DECLARE SUB aknarajz ()
DEFINT A-Z

TYPE aknahely
   szin AS INTEGER
   tolt AS INTEGER
   fal AS INTEGER
   kapcs AS INTEGER
   END TYPE

CONST elemek = 12
CONST aknax = 32, aknay = 20
CONST kocka = 10
CONST dbext = 3

DIM SHARED akna(1 TO aknax, 1 TO aknay) AS aknahely
DIM SHARED raktar(1 TO elemek, 1 TO 4) AS STRING
DIM SHARED ply, pont, zus, zuk, ext
DIM SHARED jel(0 TO 255) AS STRING
DIM SHARED queue AS STRING
DIM SHARED prg(0 TO 9) AS STRING
```

```
DIM SHARED par(1 TO 10) AS STRING, mpar
DIM SHARED billk, billn
DIM SHARED uzemmod

uzemmod = 1
inic
SELECT CASE uzemmod
CASE 0: aknarajz: mozgas 0, 0, 0, 0
CASE 1: replay "431jo.flm"
END SELECT


DATA kbkjjkfk, kfkllkjk, kjkbbklk, klkffkbk
DATA kbkjjkjk, kfkllklk, kbkjjkjk, kfkllklk
DATA kjkblkbk, klkfbkfk, kbkjfkjk, kfkljklk
DATA kbkjlkjk, kfklbklk, kjkbfkbk, klkfjkfk
DATA kbkjfkljk, kfkljkblk, kbkjlkfjk, kfklbkjlk
DATA kjklkbk, kjklkbk, kjklkbk, kjklkbk
DATA kjkbbkfk, klkffkjk, kbkjjklk, kfkllkbk
DATA k, k, k, k
DATA kjk, klk, kjk, klk
DATA kbkjjk, kfkllk, kbkjjk, kfkllk
DATA kfkljk, kjkblk, kbkjlk, kbkjfk
DATA x, x, x, x

DATA 0,"d8r4u8l4",1,"br4d8",2,"r4d4l4d4r4",3,"r4d4nl4d4l4",4,"d4r4nu4d4"
DATA 5,"nr4d4r4d4l4",6,"nr4d8r4u4l4",7,"r4d8",8,"r4d8l4u4nr4u4",9,"nr4d4r4nu4d4l4"
DATA "",""

SUB aknarajz
FOR y = 1 TO aknay
FOR x = 1 TO aknax
IF akna(x, y).fal THEN
   kockarajz x, y, 256, 0
   ELSE
   IF akna(x, y).szin THEN kockarajz x, y, akna(x, y).szin, akna(x, y).tolt
   END IF

SELECT CASE akna(x, y).tolt
CASE 1, 2: racsrajz x, y
END SELECT

IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
NEXT: NEXT
END SUB

SUB elemrajz (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": kockarajz x, y, szin(e), 0
CASE "x": extrarajz x, y
```

```
END SELECT
NEXT
END SUB

SUB elemtorles (e, i, ex, ey)
x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k", "x"
   kockatorles x, y
   IF akna(x, y).tolt THEN racsrajz x, y
   IF akna(x, y).kapcs THEN kapcsrajz x, y, akna(x, y).kapcs
END SELECT
NEXT
END SUB

SUB exec (cmd$)
SELECT CASE UCASE$(cmd$)
CASE "COPY": x1 = VAL(par(1)): y1 = VAL(par(2))
   x2 = VAL(par(3)): y2 = VAL(par(4))
   xn = VAL(par(5)): yn = VAL(par(6))
   IF x1 > x2 THEN SWAP x1, x2
   IF y1 > y2 THEN SWAP y1, y2
   FOR x = 0 TO x2 - x1: FOR y = 0 TO y2 - y1
   IF filter(par(7), x1 + x, y1 + y) THEN
      akna(xn + x, yn + y) = akna(x1 + x, y1 + y)
      kockatorles xn + x, yn + y
      END IF
   NEXT: NEXT
CASE "FILL": x1 = VAL(par(2)): y1 = VAL(par(3))
   x2 = VAL(par(4)): y2 = VAL(par(5))
   IF x1 > x2 THEN SWAP x1, x2
   IF y1 > y2 THEN SWAP y1, y2
   FOR xx = 0 TO x2 - x1: FOR yy = 0 TO y2 - y1
   x = x1 + xx: y = y1 + yy
   IF filter(par(6), x, y) THEN
      SELECT CASE VAL(par(1))
      CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
         kockatorles x, y
      CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
         kockatorles x, y: racsrajz x, y
      CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
         kockatorles x, y: racsrajz x, y
      CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
         kockatorles x, y: kockarajz x, y, 256, 0
      CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
         kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
      END SELECT
      END IF
   NEXT: NEXT
CASE "PUT": x = VAL(par(2)): y = VAL(par(3))
   SELECT CASE VAL(par(1))
```

```
   CASE 0: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 0
      kockatorles x, y
   CASE 1: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 1
      kockatorles x, y: racsrajz x, y
   CASE 2: akna(x, y).fal = 0: akna(x, y).szin = 0: akna(x, y).tolt = 2
      kockatorles x, y: racsrajz x, y
   CASE 3: akna(x, y).fal = 1: akna(x, y).szin = 0: akna(x, y).tolt = 0
      kockatorles x, y: kockarajz x, y, 256, 0
   CASE 4: akna(x, y).fal = 0: akna(x, y).szin = 17: akna(x, y).tolt = 0
      kockatorles x, y: kockarajz x, y, akna(x, y).szin, 0
   END SELECT
END SELECT
END SUB

SUB extra (e, i, ex, ey)
IF ey < aknay - 1 THEN
   SELECT CASE ext
   CASE 1
      akna(ex, ey + 1).szin = 0
      kockatorles ex, ey
      kockatorles ex, ey + 1
      aknarajz
      racs
      pontozas 0
   CASE 2
      DO
      akna(ex, ey + 1).szin = 0
      kockatorles ex, ey
      kockatorles ex, ey + 1
      ey = ey + 1
      LOOP WHILE akna(ex, ey + 1).szin
      aknarajz
      racs
      pontozas 0
   CASE 3: erg = 100: kockatorles ex, ey: ey = ey + 1
      ko$ = CHR$(ex) + CHR$(ey): m = 3
      DO
      extra3ko ko$, ex - 1, ey
      extra3ko ko$, ex + 1, ey
      extra3ko ko$, ex, ey - 1
      extra3ko ko$, ex, ey + 1
      erg = erg - 1
      IF m > LEN(ko$) THEN EXIT DO
      ex = ASC(MID$(ko$, m)): ey = ASC(MID$(ko$, m + 1))
      m = m + 2
      LOOP UNTIL erg = 0

      FOR m = 1 TO LEN(ko$) STEP 2
      x = ASC(MID$(ko$, m)): y = ASC(MID$(ko$, m + 1))
      akna(x, y).szin = 0
      kockatorles x, y
      NEXT
      aknarajz
      racs
      pontozas 0
   END SELECT
   ELSE
   kockatorles ex, ey
```

```
   racsrajz ex, ey
   END IF
END SUB

SUB extra3ko (ko$, x, y)
IF akna(x, y).szin THEN
   c$ = CHR$(x) + CHR$(y)
   i = INSTR(ko$, c$)
   IF i / 2 = i \ 2 THEN ko$ = ko$ + c$
   END IF
END SUB

SUB extrarajz (x, y)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE ext
CASE 1
   CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8
   PAINT (px + kocka / 2, py + kocka / 2), 8
CASE 2
   CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 2 - 1, 8, , , 2
   PAINT (px + kocka / 2, py + kocka / 2), 8
CASE 3
   CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , 2
   CIRCLE (px + kocka / 2, py + kocka / 2), kocka / 3, 8, , , .5
   PAINT (px + kocka / 2, py + kocka / 2), 8
END SELECT
END SUB

SUB film (f)
IF uzemmod THEN EXIT SUB

OPEN "tetris.flm" FOR APPEND AS 3
SELECT CASE f
CASE 0: PRINT #3, "0"
CASE 1
   PRINT #3, "Q";
   FOR a = 1 TO LEN(queue)
   PRINT #3, STR$(ASC(MID$(queue, a)));
   NEXT
   PRINT #3,
CASE 2: PRINT #3, "X"; STR$(ext)
CASE 3: IF billk THEN PRINT #3, "M"; STR$(billk); STR$(billn)
END SELECT
CLOSE 3
END SUB

FUNCTION filter (f$, xx, yy)
x = xx: y = yy
FOR a = 1 TO LEN(f$)
x$ = MID$(f$, a, 1)
SELECT CASE x$
CASE "x": x = x - 1
CASE "X": x = x + 1
CASE ".": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x,
y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "*": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x,
y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
```

```
CASE "/": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "#": IF akna(x, y).szin = 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 1 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "B", "b": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 0 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "G", "g": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 1 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
CASE "R", "r": IF akna(x, y).szin <> 0 AND akna(x, y).tolt = 2 AND akna(x, y).fal = 0 AND akna(x, y).kapcs = 0 THEN filter = 1: EXIT FUNCTION
END SELECT
NEXT
END FUNCTION

SUB inic
FOR y = 1 TO aknay - 1
akna(1, y).fal = 1
akna(aknax, y).fal = 1
NEXT
FOR x = 1 TO aknax
akna(x, aknay).fal = 1
NEXT

FOR e = 1 TO elemek
FOR i = 1 TO 4
READ raktar(e, i)
NEXT: NEXT

RANDOMIZE TIMER
SCREEN 13
IF uzemmod = 0 THEN ply = 1: palya ply

paletta 1, 0, 30, 0
paletta 2, 30, 0, 0
paletta 3, 30, 30, 30
paletta 4, 30, 60, 30
paletta 5, 60, 30, 30
paletta 6, 60, 20, 20
paletta 7, 20, 20, 25
paletta 8, 63, 0, 0

FOR f = -1 TO 1
paletta 0 * 3 + f + 17, 53 + f * 10, 0, 0
paletta 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10, 20 + f * 10
paletta 2 * 3 + f + 17, 0, 53 + f * 10, 0
paletta 3 * 3 + f + 17, 0, 0, 53 + f * 10
paletta 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10, 0
paletta 5 * 3 + f + 17, 40 + f * 10, 0, 40 + f * 10
paletta 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 0
paletta 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10
paletta 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10, 40 + f * 10
paletta 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10, 53 + f * 10
paletta 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10, 0

z = 20
paletta 33 + 0 * 3 + f + 17, 53 + f * 10, z, 0
paletta 33 + 1 * 3 + f + 17, 20 + f * 10, 20 + f * 10 + z, 20 + f * 10
paletta 33 + 2 * 3 + f + 17, 0, 53 + f * 10 - z, 0
```

```
paletta 33 + 3 * 3 + f + 17, 0, z, 53 + f * 10
paletta 33 + 4 * 3 + f + 17, 40 + f * 10, 20 + f * 10 + z, 0
paletta 33 + 5 * 3 + f + 17, 40 + f * 10, z, 40 + f * 10
paletta 33 + 6 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 0
paletta 33 + 7 * 3 + f + 17, 53 + f * 10, z, 53 + f * 10
paletta 33 + 8 * 3 + f + 17, 40 + f * 10, 40 + f * 10 + z, 40 + f * 10
paletta 33 + 9 * 3 + f + 17, 53 + f * 10 - z, 53 + f * 10, 53 + f * 10 - z
paletta 33 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 + z, 0

p = 20
paletta 66 + 0 * 3 + f + 17, 53 + f * 10 - p, 0, 0
paletta 66 + 1 * 3 + f + 17, 20 + f * 10 + p, 20 + f * 10, 20 + f * 10
paletta 66 + 2 * 3 + f + 17, p, 53 + f * 10, 0
paletta 66 + 3 * 3 + f + 17, p, 0, 53 + f * 10
paletta 66 + 4 * 3 + f + 17, 40 + f * 10 + p, 20 + f * 10, 0
paletta 66 + 5 * 3 + f + 17, 40 + f * 10 + p, 0, 40 + f * 10
paletta 66 + 6 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 0
paletta 66 + 7 * 3 + f + 17, 53 + f * 10, 0, 53 + f * 10 - p
paletta 66 + 8 * 3 + f + 17, 40 + f * 10 + p, 40 + f * 10, 40 + f * 10
paletta 66 + 9 * 3 + f + 17, 53 + f * 10, 53 + f * 10 - p, 53 + f * 10 - p
paletta 66 + 10 * 3 + f + 17, 53 + f * 10, 31 + f * 10 - p, 0
NEXT

DO
READ kod$, rajz$
IF kod$ = "" THEN EXIT DO
jel(ASC(kod$)) = rajz$
LOOP
END SUB

SUB iras (t$, x, y, c)
FOR a = 1 TO LEN(t$)
x$ = MID$(t$, a, 1)
j$ = jel$(ASC(x$))
DRAW "BM" + LTRIM$(STR$(x)) + "," + LTRIM$(STR$(y)) + "C" + LTRIM$(STR$(c)) + j$
x = x + 6
NEXT
END SUB

SUB kapcsolo (be$)
FOR a = 1 TO LEN(be$)
k = ASC(MID$(be$, a, 1))
program prg(k)
NEXT
END SUB

SUB kapcsrajz (x, y, k)
px = (x - 1) * kocka: py = (y - 1) * kocka
LINE (px, py)-STEP(9, 9), 8, B
END SUB

SUB kockarajz (x, y, szin, t)
px = (x - 1) * kocka: py = (y - 1) * kocka
SELECT CASE szin
CASE 256
   LINE (px, py)-STEP(3, 3), 6, BF
   LINE (px + 5, py)-STEP(3, 3), 6, BF
   LINE (px + 2, py + 5)-STEP(3, 3), 6, BF
```

```
      LINE (px + 7, py + 5)-STEP(2, 3), 6, BF
      LINE (px, py + 5)-STEP(0, 3), 6
      LINE (px, py + 4)-STEP(9, 0), 7
      LINE (px, py + 9)-STEP(9, 0), 7
      LINE (px + 4, py)-STEP(0, 3), 7
      LINE (px + 9, py)-STEP(0, 3), 7
      LINE (px + 1, py + 5)-STEP(0, 3), 7
      LINE (px + 6, py + 5)-STEP(0, 3), 7
   CASE ELSE
      LINE (px + 1, py + 1)-STEP(7, 7), szin + t * 33, BF
      IF szin <> 0 THEN LINE (px, py)-STEP(9, 9), 3 + t, B ELSE LINE (px, py)-STEP(9, 9), 0, B
   END SELECT
   END SUB

   SUB kockatorles (x, y)
   px = (x - 1) * kocka: py = (y - 1) * kocka
   LINE (px, py)-STEP(9, 9), 0, BF
   END SUB

   FUNCTION mehet (e, i, ex, ey)
   x = ex: y = ey: mehet = 1
   ceruza$ = raktar(e, i)
   FOR c = 1 TO LEN(ceruza$)
   p$ = MID$(ceruza$, c, 1)
   SELECT CASE p$
   CASE "b": x = x - 1
   CASE "j": x = x + 1
   CASE "f": y = y - 1
   CASE "l": y = y + 1
   CASE "k", "x": IF akna(x, y).szin OR akna(x, y).fal = 1 THEN mehet = SGN(x - ex) + 3 - (x - ex
= 2): EXIT FUNCTION
   END SELECT
   NEXT
   END FUNCTION

   SUB mozgas (e, i, x, y)
   IF uzemmod = 0 THEN ujelem e, i, x, y
   DO
   elemrajz e, i, x, y

   SELECT CASE uzemmod
   CASE 0
      a$ = INKEY$
      IF LEN(a$) = 2 THEN
         k = INSTR("KMHP", RIGHT$(a$, 1))
         IF k THEN
            billn = billn + 1
            IF k <> billk THEN film 3: billn = 0
            billk = k
            END IF
         END IF
   CASE 1
      a$ = INPUT$(1)
      a$ = CHR$(0) + MID$("KMHP", billk, 1)
   END SELECT

   SELECT CASE a$
   CASE CHR$(0) + "K"
```

```
    IF mehet(e, i, x - 1, y) = 1 THEN
      elemtorles e, i, x, y: x = x - 1
      ELSE
      IF raktar(e, i) = "x" AND akna(x - 1, y).fal = 0 THEN
        akna(x - 1, y).szin = 0
        kockatorles x - 1, y
        kockatorles x, y
        racsrajz x - 1, y
        racsrajz x, y
        ujelem e, i, x, y
        END IF
      END IF
  CASE CHR$(0) + "M"
    IF mehet(e, i, x + 1, y) = 1 THEN
      elemtorles e, i, x, y: x = x + 1
      ELSE
      IF raktar(e, i) = "x" AND akna(x + 1, y).fal = 0 THEN
        akna(x + 1, y).szin = 0
        kockatorles x + 1, y
        kockatorles x, y
        racsrajz x + 1, y
        racsrajz x, y
        ujelem e, i, x, y
        END IF
      END IF
  CASE CHR$(0) + "H": ii = i + 1: IF ii = 5 THEN ii = 1
    SELECT CASE mehet(e, ii, x, y)
    CASE 1: elemtorles e, i, x, y: i = ii
    CASE 2: IF mehet(e, ii, x + 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x + 1
    CASE 4
      IF mehet(e, ii, x - 1, y) = 1 THEN
        elemtorles e, i, x, y: i = ii: x = x - 1
        ELSE
        IF mehet(e, ii, x - 2, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
        END IF
    CASE 5: IF mehet(e, ii, x - 1, y) = 1 THEN elemtorles e, i, x, y: i = ii: x = x - 2
    END SELECT
  CASE CHR$(0) + "P"
    IF mehet(e, i, x, y + 1) = 1 THEN
      elemtorles e, i, x, y: y = y + 1
      ELSE
      IF uzemmod = 0 THEN billn = billn + 1: film 3: billk = 0
      ragadas e, i, x, y: ujelem e, i, x, y
      END IF
  END SELECT
  LOOP UNTIL uzemmod
  END SUB

  SUB paletta (p, r, g, B)
  OUT &H3C8, p
  OUT &H3C9, r
  OUT &H3C9, g
  OUT &H3C9, B
  END SUB

  SUB palya (ply)
  IF ply THEN
    OPEN "tetris.lev" FOR INPUT AS 2
```

```
   OPEN "tetris.flm" FOR APPEND AS 3
   END IF
FOR a = 0 TO 9: prg(a) = "": NEXT
DO
IF uzemmod = 0 THEN LINE INPUT #2, l$
IF VAL(l$) = ply AND INSTR(l$, ":") = 0 OR ply = 0 THEN
   FOR y = 1 TO aknay - 1
   LINE INPUT #2, l$
   IF ply THEN PRINT #3, l$
   FOR x = 2 TO aknax - 1
   x$ = MID$(l$, x - 1, 1)
   IF akna(x, y).tolt + akna(x, y).fal THEN kockatorles x, y
   akna(x, y).tolt = 0: akna(x, y).fal = 0: akna(x, y).kapcs = 0
   SELECT CASE x$
   CASE "*": akna(x, y).tolt = 1
   CASE "/": akna(x, y).tolt = 2
   CASE "#": akna(x, y).fal = 1: akna(x, y).szin = 0
   CASE "1" TO "9": akna(x, y).kapcs = VAL(x$)
   END SELECT
   NEXT: NEXT

   DO UNTIL EOF(2)
   LINE INPUT #2, l$
   IF ply = 0 AND l$ = "0" THEN EXIT DO
   IF MID$(l$, 2, 1) = ":" THEN
      i = VAL(l$)
      prg(i) = LTRIM$(MID$(l$, 3))
      IF ply THEN PRINT #3, l$
      ELSE
      IF VAL(l$) THEN EXIT DO
      prg(i) = prg(i) + ":" + l$
      IF ply THEN PRINT #3, l$
      END IF
   LOOP
   IF ply THEN CLOSE 2, 3: film 0
   EXIT SUB
   END IF
LOOP UNTIL EOF(2)
IF ply THEN CLOSE 2, 3: film 0
END SUB

SUB pontozas (p)
p = INT((p / 2) + .5) * 2
pont = pont + p
aknarajz
iras STR$(pont), 319 - LEN(STR$(pont)) * 6, 191, 15
END SUB

SUB program (pr$)
s$ = pr$
DO UNTIL s$ = ""
i = INSTR(s$ + ":", ":")
p$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
cmd$ = word$(p$)
mpar = 1
DO UNTIL p$ = ""
par(mpar) = word$(p$)
```

```
mpar = mpar + 1
LOOP
mpar = mpar - 1
exec cmd$
LOOP
END SUB

SUB queuerajz
LINE (2, 191)-(100, 197), 0, BF
LINE (92, 191)-(100, 197), 1, BF
FOR q = 1 TO LEN(queue) STEP 2
qe = ASC(MID$(queue, q))
qi = ASC(MID$(queue, q + 1))
x = 100 - q * 5
y = 193
ceruza$ = raktar(qe, qi)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 2
CASE "j": x = x + 2
CASE "f": y = y - 2
CASE "l": y = y + 2
CASE "k": LINE (x, y)-STEP(1, 1), szin(qe), B
CASE "x": CIRCLE (x, y), 1, 8
END SELECT
NEXT: NEXT
END SUB

SUB racs
FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN IF akna(x, y).szin = 0 THEN EXIT SUB
IF akna(x, y).tolt = 2 THEN IF akna(x, y).szin <> 0 THEN EXIT SUB
NEXT: NEXT

FOR x = 2 TO aknax - 1
FOR y = 1 TO aknay - 1
IF akna(x, y).tolt = 1 THEN
   akna(x, y).szin = 0
   kockatorles x, y
   pontozas 10
   END IF
NEXT: NEXT

ply = ply + 1: IF uzemmod = 0 THEN palya ply
aknarajz
pontozas 0
END SUB

SUB racsrajz (x, y)
IF akna(x, y).szin = 0 THEN
   px = (x - 1) * kocka: py = (y - 1) * kocka
   LINE (px, py)-STEP(9, 9), akna(x, y).tolt, BF
   END IF
END SUB

SUB ragadas (e, i, ex, ey)
```

```
IF raktar(e, i) = "x" THEN extra e, i, ex, ey: EXIT SUB

x = ex: y = ey
ceruza$ = raktar(e, i)
FOR c = 1 TO LEN(ceruza$)
p$ = MID$(ceruza$, c, 1)
SELECT CASE p$
CASE "b": x = x - 1
CASE "j": x = x + 1
CASE "f": y = y - 1
CASE "l": y = y + 1
CASE "k": akna(x, y).szin = szin(e)
   IF akna(x, y).kapcs THEN be$ = be$ + CHR$(akna(x, y).kapcs)
END SELECT
NEXT
pontozas 2
kapcsolo be$
zus = 0: zuk = 0
sorok
pontozas zus * zuk
racs
END SUB

SUB replay (f$)
OPEN f$ FOR INPUT AS 2
palya 0
aknarajz

DO UNTIL EOF(2)
LINE INPUT #2, l$
SELECT CASE word$(l$)
CASE "M": billk = VAL(word$(l$)): billn = VAL(word$(l$))
   DO: mozgas e, i, x, y: billn = billn - 1: LOOP UNTIL billn = 0
CASE "Q": queue = ""
   DO UNTIL l$ = "": queue = queue + CHR$(VAL(word$(l$))): LOOP
   queuerajz
   e = ASC(queue): i = ASC(MID$(queue, 2)): x = aknax \ 2: y = 2
CASE "X": ext = VAL(word$(l$))
END SELECT
IF ply THEN ply = 0: palya 0: aknarajz
LOOP
END SUB

SUB sorok
FOR y = 1 TO aknay - 1
tele = 1: k = 2: vk = 0
FOR x = 2 TO aknax - 1
SELECT CASE akna(x, y).fal
CASE 0: IF akna(x, y).szin = 0 THEN tele = 0: vk = 0 ELSE vk = 1
CASE 1
   IF tele = 1 THEN
      IF akna(x - 1, y).fal = 0 THEN IF vk THEN zuditas y, k, x - 1: k = x + 1
      ELSE tele = 1: k = x + 1: vk = 0
      END IF
END SELECT
NEXT
IF tele THEN IF vk THEN zuditas y, k, x - 1
NEXT
```

```
END SUB

FUNCTION szin (e)
szin = e * 3 + 14
END FUNCTION

SUB ujelem (e, i, x, y)
DO WHILE LEN(queue) < 20
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
LOOP

e = ASC(queue)
i = ASC(MID$(queue, 2))
queue = MID$(queue, 3)
queue = queue + CHR$(INT(RND * elemek + 1)) + CHR$(INT(RND * 4 + 1))
film 1

queuerajz

x = aknax \ 2
y = 2
ext = INT(RND * dbext) + 1
film 2
END SUB

FUNCTION word$ (s$)
s$ = LTRIM$(s$)
i = INSTR(s$ + " ", " ")
word$ = LEFT$(s$, i - 1)
s$ = MID$(s$, i + 1)
END FUNCTION

SUB zuditas (y, k, v)
zus = zus + 1
zuk = zuk + v - k + 1
DIM f(1 TO aknax)

csf = 1
FOR yy = y TO 2 STEP -1
FOR x = k TO v
IF akna(x, yy).fal THEN f(x) = 1 ELSE csf = 0
IF f(x) = 0 THEN
   akna(x, yy).szin = akna(x, yy - 1).szin
   kockatorles x, yy
   END IF
NEXT: NEXT

FOR x = k TO v
IF f(x) = 0 THEN
   akna(x, 1).szin = 0
   kockatorles x, 1
   END IF
NEXT
aknarajz
IF csf = 0 THEN sorok
END SUB
```

```
..............................
..............................
..............................
..............................
..............................
..............................
..........#********#..........
...///....#********#....///...
.../1/....#********#..../2/...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
...///....#********#....///...
1: put 0 12 7:put 0 12 8:put 0 12 9:put 0 12 10:put 0 12 11
put 0 12 12:put 0 12 13:put 0 12 14:put 0 12 15:put 0 12 16
put 0 12 17:put 0 12 18:put 0 12 19
2: put 3 12 7:put 3 12 8:put 3 12 9:put 3 12 10:put 3 12 11
put 3 12 12:put 3 12 13:put 3 12 14:put 3 12 15:put 3 12 16
put 3 12 17:put 3 12 18:put 3 12 19
0
Q 11 4 2 1 7 4 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2
X 3
M 4 8
M 2 5
M 4 10
Q 2 1 7 4 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1
X 1
M 1 12
M 3 1
M 4 16
Q 7 4 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4
X 2
M 1 13
M 4 7
M 2 1
M 4 6
Q 7 2 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2
X 1
M 1 12
M 4 5
M 3 2
M 4 5
Q 6 2 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4
X 2
M 4 12
M 2 1
M 4 5
Q 10 1 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4
X 1
M 1 11
M 4 8
Q 9 2 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3
X 2
```

M 4 13
M 3 1
M 1 1
M 4 5
Q 5 3 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3
X 1
M 2 3
M 4 16
Q 3 1 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4
X 1
M 1 2
M 4 17
Q 5 2 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4
X 2
M 3 1
M 4 16
Q 12 2 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2
X 1
M 1 11
M 4 7
Q 3 1 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1
X 2
M 1 1
M 4 9
M 2 1
M 4 5
Q 10 4 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1
X 3
M 3 1
M 2 3
M 4 14
M 1 1
M 4 1
Q 6 2 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4
X 2
M 4 8
M 2 2
M 4 5
Q 9 4 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2
X 2
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 11
Q 1 4 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2
X 1
M 3 3
M 4 13
Q 8 3 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2
X 3
M 1 3
M 4 17
Q 10 3 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2
X 1
M 3 1

```
M 1 2
M 4 15
Q 2 4 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2
X 2
M 1 4
M 4 5
M 2 1
M 4 9
Q 4 4 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2
X 1
M 4 6
M 1 2
M 4 6
Q 2 2 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1
X 3
M 1 3
M 4 10
Q 6 1 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4
X 3
M 4 11
Q 11 1 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3
X 3
M 3 2
M 2 4
M 4 12
Q 4 4 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3
X 1
M 4 1
M 1 2
M 4 9
Q 7 2 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1
X 1
M 2 2
M 3 2
M 1 1
M 4 9
Q 1 2 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2
X 3
M 2 1
M 3 1
M 2 2
M 4 11
Q 1 2 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2
X 3
M 2 3
M 3 3
M 4 10
Q 7 2 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2
X 3
M 4 8
Q 2 2 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1
X 1
M 2 2
M 4 7
Q 1 2 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1
X 2
M 2 1
M 1 2
M 3 2
```

```
M 1 2
M 4 7
Q 2 1 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4
X 2
M 3 1
M 1 2
M 4 7
Q 6 4 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2
X 3
M 2 7
M 4 13
M 1 1
M 4 4
Q 12 3 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2
X 3
M 1 12
M 4 2
M 1 1
M 4 1
M 2 1
M 4 4
Q 4 3 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4
X 2
M 3 1
M 2 3
M 4 8
Q 7 1 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3
X 3
M 3 1
M 1 1
M 4 7
Q 9 2 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3
X 2
M 2 3
M 4 6
Q 11 2 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2
X 1
M 2 1
M 4 5
Q 12 2 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1
X 1
M 1 4
M 4 5
Q 11 1 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3
X 2
M 2 3
M 3 1
M 2 2
M 4 1
M 1 1
M 4 4
...............................
...............................
...............................
...............................
########/............./########
********/............./********
********/............./********
********/............./********
```

```
********/............/********
********/............/********
########/............/########
..............................
..............................
.............####.............
.............****.............
.............****.............
/////////////****/////////////
/////////////****/////////////
/////////////****/////////////
0
Q 7 1 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1
X 3
M 1 3
M 4 5
M 1 9
M 4 3
M 2 2
M 4 1
Q 11 4 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3
X 1
M 3 1
M 1 4
M 4 5
M 1 11
M 4 4
Q 10 2 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1
X 2
M 1 4
M 4 5
M 1 8
M 4 3
Q 4 2 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2
X 1
M 3 1
M 4 1
M 1 2
M 4 3
M 1 1
M 4 1
M 1 5
M 4 4
Q 2 4 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3
X 3
M 4 6
M 3 1
M 2 14
M 4 3
Q 6 3 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3
X 1
M 4 1
M 1 1
M 4 4
M 2 16
M 4 2
Q 8 3 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2
X 1
M 1 5
```

```
M 4 4
M 1 8
M 4 4
Q 2 2 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2
X 3
M 1 2
M 4 9
M 1 1
M 3 1
M 1 1
M 4 3
M 2 1
M 4 1
M 2 4
M 4 5
Q 1 1 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2
X 3
M 1 1
M 4 7
M 2 11
M 4 2
Q 7 3 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1
X 1
M 1 3
M 4 3
M 1 1
M 4 2
M 1 4
M 4 2
Q 8 1 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2
X 1
M 1 5
M 4 4
M 1 11
M 4 3
Q 6 3 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2
X 2
M 1 1
M 4 7
M 1 2
M 4 7
M 2 2
M 4 2
Q 12 1 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1
X 3
M 4 8
M 2 7
M 4 8
Q 1 2 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1
X 3
M 1 2
M 3 1
M 4 5
M 2 10
M 4 1
M 2 2
M 1 1
M 4 2
Q 11 3 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3
```

```
X 2
M 1 5
M 4 4
M 1 9
M 4 2
Q 10 3 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1
X 2
M 1 2
M 4 7
M 1 1
M 4 6
M 2 4
M 3 1
M 4 1
M 3 1
M 4 2
Q 9 2 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3
X 2
M 1 3
M 3 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 5
M 4 3
Q 8 2 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2
X 2
M 1 2
M 4 5
M 2 15
M 4 3
Q 11 2 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2
X 1
M 1 1
M 4 5
M 2 13
M 4 2
Q 12 1 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4
X 1
M 2 1
M 4 4
Q 12 2 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4
X 1
M 2 2
M 4 4
Q 9 2 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4
X 3
M 3 1
M 1 3
M 4 3
M 1 1
M 4 1
M 1 11
M 4 1
Q 6 1 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3
```

```
X 2
M 1 4
M 4 4
M 1 9
M 4 1
Q 10 1 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4
X 2
M 3 1
M 4 6
M 2 3
M 4 1
M 2 1
M 4 7
M 1 2
M 4 2
Q 2 3 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1
X 1
M 4 5
M 2 14
M 4 1
Q 10 1 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1
X 1
M 2 3
M 3 1
M 2 3
M 4 3
M 2 1
M 4 1
M 3 1
M 2 9
M 4 1
Q 3 3 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4
X 1
M 1 3
M 4 14
M 2 3
M 4 1
Q 2 2 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4
X 1
M 3 1
M 4 5
M 2 9
M 4 2
Q 8 2 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2
X 1
M 4 2
M 2 6
M 4 1
M 2 1
M 4 1
M 2 2
M 1 2
M 4 7
M 1 4
M 4 2
M 1 1
M 4 1
Q 8 4 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3
X 3
```

M 4 6
M 1 1
M 4 1
M 1 1
M 4 6
M 2 1
M 4 1
Q 4 4 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2
X 3
M 3 1
M 4 8
M 2 1
M 4 3
Q 11 4 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2
X 2
M 3 2
M 2 6
M 4 4
M 2 7
M 4 1
Q 11 3 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4
X 3
M 3 1
M 2 6
M 4 1
M 2 1
M 4 4
M 2 5
M 4 1
Q 4 4 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4
X 2
M 3 1
M 1 2
M 4 10
M 2 1
M 4 1
Q 2 1 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1
X 2
M 1 3
M 4 4
M 1 7
M 4 2
Q 4 1 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3
X 1
M 4 6
M 2 1
M 4 3
Q 5 4 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2
X 1
M 1 1
M 3 1
M 1 1
M 4 9
M 2 1
M 4 1
Q 12 4 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1
X 1
M 2 5
M 4 4

Q 3 2 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1
X 1
M 4 2
M 2 1
M 4 1
M 2 1
M 4 5
Q 8 3 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2
X 1
M 4 2
M 1 3
M 4 1
M 2 9
M 4 1
M 2 3
M 4 1
Q 5 2 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4
X 1
M 1 2
M 4 8
Q 4 2 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2
X 3
M 2 6
M 4 7
M 3 1
M 1 2
M 4 4
M 1 1
M 4 1
Q 2 4 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1
X 3
M 1 1
M 3 1
M 1 4
M 4 4
M 1 6
M 4 1
Q 2 4 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4
X 1
M 2 3
M 4 8
Q 10 1 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4
X 1
M 4 1
M 3 1
M 4 6
Q 11 3 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4
X 2
M 1 1
M 4 6
Q 5 2 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1
X 2
M 3 1
M 2 2
M 4 6
Q 10 1 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2
X 2
M 2 3
M 3 1

```
M 2 2
M 4 6
M 1 1
M 4 4
Q 3 1 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1
X 1
M 2 3
M 4 4
Q 9 2 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2
X 1
M 2 7
M 4 4
M 2 1
M 4 1
..............................
..............................
..............................
..............................
..............................
#######................#######
******#................#******
******#................#******
5*****#................#*****6
******#................#******
******#................#******
******#................#******
#######................#######
******#................#******
******#.......12.......#******
7*****#.......34.......#*****8
******#................#******
******#................#******
******#...9............#******
1: put 2 8 9: put 2 8 10
2: put 2 8 16: put 2 8 17
3: put 2 25 16: put 2 25 17
4: put 2 25 9: put 2 25 10
5: put 3 8 16: put 3 8 17: put 2 8 15: put 2 8 14
6: put 3 25 16: put 3 25 17: put 2 25 15: put 2 25 14
7: put 3 25 9: put 3 25 10: put 2 25 8: put 2 25 7
8: put 3 8 9: put 3 8 10: put 2 8 8: put 2 8 7
9: copy 8 6 8 19 9 6 *bg/r#: fill 0 8 6 8 19 x.*bg/r
copy 25 6 25 19 24 6 *bg/r#: fill 0 25 6 25 19 X.*bg/r
0
Q 7 4 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4
X 2
M 1 4
M 3 1
M 4 13
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 1 3
M 4 3
Q 2 2 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2
X 3
M 2 5
```

```
M 4 12
M 3 1
M 4 2
M 1 3
M 4 1
M 1 1
M 4 3
Q 10 1 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4
X 2
M 2 6
M 4 11
M 1 1
M 4 1
M 1 1
M 4 1
M 1 1
M 4 1
M 1 4
M 4 1
M 1 1
M 4 2
Q 7 4 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1
X 2
M 2 7
M 3 1
M 4 13
M 1 4
M 4 1
M 1 1
M 4 3
Q 9 4 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3
X 2
M 1 3
M 4 11
M 3 1
M 4 2
M 2 2
M 4 3
Q 7 4 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3
X 2
M 1 5
M 3 3
M 4 9
M 2 1
M 4 3
M 2 7
M 1 2
M 4 3
Q 8 1 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4
X 3
M 1 3
M 4 13
M 2 1
M 4 3
Q 1 2 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4
X 3
M 3 1
M 1 4
M 4 12
```

```
M 2 5
M 1 1
M 4 2
Q 4 1 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2
X 3
M 2 6
M 4 1
M 2 1
M 4 6
M 2 8
M 4 3
Q 3 2 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4
X 1
M 3 1
M 1 4
M 4 2
M 1 1
M 4 1
M 1 1
M 4 5
M 1 7
M 4 3
Q 7 4 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2
X 2
M 1 5
M 3 1
M 4 15
M 1 11
M 4 3
Q 3 2 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4
X 2
M 2 7
M 4 1
M 3 1
M 4 7
M 2 7
M 3 1
M 4 1
Q 7 4 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2
X 1
M 2 3
M 3 1
M 2 3
M 4 1
M 2 1
M 4 7
M 2 4
M 4 2
M 2 1
M 4 1
Q 5 1 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2
X 1
M 2 5
M 3 1
M 2 1
M 3 1
M 4 1
M 2 1
M 4 6
```

M 2 6
M 4 2
Q 2 3 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3
X 3
M 1 5
M 4 1
M 1 1
M 4 6
M 1 6
M 3 2
M 1 1
M 3 1
M 2 2
M 3 2
M 2 5
M 4 7
M 1 7
M 3 3
M 2 2
M 3 1
M 1 1
M 4 3
Q 10 3 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4
X 1
M 1 3
M 3 1
M 1 2
M 4 3
M 3 1
M 4 2
M 1 1
M 4 2
M 1 7
M 3 1
M 1 1
M 4 2
Q 5 4 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2
X 2
M 1 4
M 3 1
M 1 2
M 4 13
M 1 7
M 3 1
M 1 1
M 4 2
Q 4 4 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3
X 1
M 2 6
M 4 11
M 3 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 7
M 4 3
Q 3 2 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2

```
X 1
M 1 4
M 4 2
M 1 1
M 4 4
M 3 1
M 4 2
M 1 5
M 4 2
M 1 1
M 4 1
Q 1 4 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1
X 3
M 2 7
M 4 1
M 3 1
M 4 14
M 2 5
M 1 1
M 3 1
M 4 1
M 2 1
M 4 1
Q 1 2 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4
X 2
M 2 1
M 1 3
M 2 8
M 4 14
M 3 2
M 1 1
M 4 1
M 1 1
M 4 2
Q 5 4 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4
X 2
M 1 4
M 4 9
M 3 2
M 4 3
M 2 1
M 4 3
Q 11 2 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1
X 3
M 3 1
M 1 3
M 2 9
M 4 2
M 2 1
M 4 1
M 2 1
M 4 2
M 2 9
M 4 2
Q 3 2 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2
X 1
M 2 8
M 4 7
M 1 1
```

```
M 3 1
M 4 6
M 2 6
M 3 1
M 2 1
M 4 3
Q 5 3 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3
X 2
M 1 4
M 3 1
M 4 8
M 3 1
M 4 4
M 2 2
M 3 1
M 4 1
Q 7 4 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3
X 2
M 3 3
M 2 6
M 4 12
M 1 7
M 4 1
Q 4 2 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3
X 1
M 1 1
M 2 1
M 1 3
M 4 13
Q 4 3 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4
X 3
M 3 1
M 1 4
M 4 11
Q 7 2 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1
X 3
M 1 4
M 3 3
M 1 1
M 4 7
M 1 1
M 4 1
M 1 7
M 4 1
Q 9 1 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4
X 1
M 2 6
M 3 1
M 4 1
M 2 1
M 4 11
M 2 11
M 4 2
Q 5 4 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1
X 3
M 3 1
M 2 5
M 3 1
M 2 1
```

M 4 13
M 1 3
M 4 1
Q 12 4 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2
X 2
M 2 3
M 4 3
Q 1 1 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3
X 2
M 1 6
M 4 15
M 2 1
M 4 3
Q 1 2 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2
X 2
M 2 5
M 3 3
M 2 1
M 4 18
Q 6 3 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4
X 3
M 2 6
M 4 1
M 1 1
M 4 15
Q 4 3 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2
X 1
M 2 5
M 1 9
M 3 1
M 1 2
M 4 15
M 2 1
M 4 1
Q 6 3 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4
X 3
M 1 6
M 4 5
M 1 9
M 4 1
Q 5 4 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2
X 1
M 1 2
M 3 1
M 1 1
M 3 1
M 1 1
M 4 9
Q 9 1 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4
X 1
M 2 2
M 3 1
M 2 1
M 4 17
Q 1 4 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2
X 1
M 3 5
M 2 6
M 4 6

M 2 6
M 4 2
Q 8 1 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1
X 1
M 1 6
M 4 18
Q 5 2 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1
X 2
M 1 6
M 4 17
Q 9 3 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1
X 3
M 2 2
M 3 1
M 2 1
M 4 17
Q 3 2 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1
X 1
M 2 6
M 4 12
M 3 1
M 4 4
M 1 1
M 4 1
Q 4 4 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3
X 2
M 2 4
M 1 3
M 2 1
M 1 1
M 4 6
Q 4 2 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2
X 2
M 3 2
M 1 2
M 2 8
M 4 11
M 3 1
M 4 1
M 2 4
M 4 5
Q 1 4 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1
X 3
M 2 5
M 3 1
M 2 1
M 4 14
M 1 1
M 4 1
Q 9 2 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2
X 3
M 2 3
M 4 15
Q 6 4 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1
X 1
M 1 1
M 4 2
M 1 5
M 4 3

M 1 6
M 4 2
Q 10 2 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3
X 1
M 2 3
M 4 13
Q 1 1 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3
X 3
M 1 1
M 4 7
Q 6 1 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2
X 2
M 2 5
M 4 9
M 2 1
M 4 2
M 2 1
M 4 1
Q 7 1 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4
X 2
M 2 1
M 3 3
M 1 5
M 3 1
M 4 6
M 1 5
M 3 3
M 2 1
M 4 3
M 1 1
M 4 1
Q 2 1 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1
X 2
M 3 1
M 1 5
M 4 14
Q 7 3 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3
X 1
M 1 5
M 4 5
M 1 8
M 4 1
Q 5 2 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4
X 3
M 2 5
M 4 8
M 1 2
M 4 2
Q 6 1 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4
X 3
M 1 2
M 4 5
Q 3 2 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4
X 2
M 3 1
M 2 1
M 4 1
M 1 4
M 3 1

M 1 1
M 4 8
Q 9 1 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2
X 3
M 3 1
M 1 6
M 4 14
Q 2 3 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3
X 1
M 3 1
M 2 3
M 4 6
Q 12 3 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4
X 1
M 2 8
M 1 1
M 4 11
Q 9 2 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1
X 2
M 1 7
M 2 1
M 4 12
Q 4 4 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1
X 2
M 3 1
M 4 4
Q 10 1 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1
X 3
M 3 1
M 2 5
M 4 13
Q 7 3 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4
X 3
M 2 6
M 4 11
Q 4 4 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1
X 2
M 2 5
M 3 1
M 2 1
M 4 7
M 1 1
M 4 2
Q 4 4 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4
X 3
M 2 5
M 3 1
M 4 2
M 2 1
M 4 3
M 2 3
M 3 2
M 2 1
M 3 1
M 1 1
M 4 1
M 3 1
M 4 4
Q 9 4 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2

X 2
M 1 5
M 4 11
Q 2 2 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3
X 2
M 1 6
M 4 10
Q 5 3 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4
X 2
M 3 1
M 2 6
M 4 1
M 2 1
M 4 10
Q 4 4 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1
X 2
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 4
M 2 4
M 4 1
M 3 1
M 4 3
Q 12 1 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1
X 3
M 1 3
M 4 10
Q 6 1 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1
X 3
M 2 5
M 4 5
M 2 7
M 1 1
M 4 4
Q 12 1 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3
X 1
M 2 5
M 4 5
M 2 5
M 4 4
Q 7 4 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1
X 3
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 3
M 2 1
M 4 1
M 2 9
M 4 3
Q 6 1 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2
X 1

M 2 5
M 4 5
M 2 10
M 4 2
Q 1 4 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1
X 1
M 1 3
M 3 1
M 1 1
M 4 1
M 1 1
M 4 2
M 2 1
M 4 9
M 1 2
M 4 1
M 1 5
M 4 2
M 1 2
M 4 1
Q 9 2 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2
X 1
M 1 4
M 4 12
M 2 1
M 4 4
Q 7 3 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1
X 2
M 3 1
M 2 5
M 4 4
M 2 1
M 4 4
M 1 1
M 4 1
Q 3 4 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4
X 2
M 1 3
M 4 15
Q 8 1 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1
X 1
M 1 2
M 4 9
M 2 1
M 4 1
M 2 1
M 4 4
Q 4 1 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3
X 1
M 1 3
M 3 1
M 2 10
M 4 9
Q 2 1 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4
X 2
M 3 1
M 1 3
M 4 7
M 1 1

M 4 7
Q 10 3 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1
X 1
M 1 3
M 3 1
M 2 8
M 4 3
M 3 1
M 4 2
M 2 5
M 4 1
M 3 1
M 4 4
Q 5 1 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4
X 1
M 3 3
M 1 3
M 4 9
M 2 1
M 4 1
M 2 1
M 4 4
Q 1 2 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3
X 1
M 2 3
M 3 3
M 2 1
M 4 5
M 2 2
M 4 1
M 2 5
M 4 1
M 3 4
M 4 2
Q 1 1 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2
X 2
M 3 2
M 2 2
M 4 2
M 2 4
M 4 3
M 2 9
M 4 2
Q 1 2 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3
X 3
M 3 1
M 1 2
M 3 3
M 1 1
M 4 12
Q 6 1 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1
X 1
M 1 3
M 4 7
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1

M 4 1
M 1 1
M 4 2
Q 8 4 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2
X 1
M 1 3
M 4 3
M 1 2
M 4 2
M 1 4
M 4 2
M 1 1
M 4 2
Q 6 1 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3
X 2
M 1 3
M 4 7
M 2 1
M 4 4
Q 1 3 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3
X 3
M 3 3
M 1 1
M 3 3
M 2 3
M 4 6
Q 3 4 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1
X 3
M 2 4
M 3 1
M 1 19
M 4 3
Q 7 1 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1
X 2
M 3 3
M 1 1
M 3 2
M 1 3
M 4 11
Q 8 4 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1
X 3
M 1 2
M 4 8
M 2 3
M 4 2
M 2 1
M 4 2
Q 4 3 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2
X 1
M 1 4
M 4 6
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 2
M 4 1

M 2 1
M 4 1
Q 2 2 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3
X 1
M 1 5
M 4 13
Q 2 3 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2
X 2
M 3 1
M 1 5
M 4 9
Q 8 1 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1
X 1
M 1 2
M 4 8
M 2 2
M 4 2
M 2 1
M 4 1
Q 7 2 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2
X 1
M 1 6
M 4 8
Q 6 3 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2
X 2
M 4 1
M 2 5
M 4 4
M 2 5
M 4 2
Q 5 3 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4
X 3
M 2 1
M 1 3
M 2 5
M 1 3
M 3 2
M 1 9
M 4 2
M 1 1
M 4 1
M 1 1
M 4 1
Q 2 1 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1
X 2
M 1 2
M 3 1
M 1 2
M 4 7
Q 12 1 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4
X 1
M 1 1
M 2 4
M 4 4
Q 8 1 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2
X 3
M 1 2
M 4 7
M 2 2

M 4 4
Q 7 2 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4
X 1
M 3 5
M 1 2
M 3 1
M 4 8
M 2 1
M 3 1
M 4 3
Q 4 3 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4
X 2
M 1 2
M 2 6
M 4 1
M 3 1
M 2 1
M 4 4
M 2 1
M 4 4
Q 8 2 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3
X 3
M 1 2
M 4 8
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
Q 6 1 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4
X 1
M 2 3
M 4 3
M 2 1
M 4 5
Q 7 2 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2
X 1
M 3 1
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 4 5
Q 6 2 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2
X 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 4
Q 6 4 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1
X 2
M 4 6
Q 11 1 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3
X 1
M 1 3
M 3 3

```
M 2 1
M 4 8
M 3 3
M 2 1
M 4 1
M 3 8
M 2 1
M 3 1
M 1 1
M 3 2
M 4 1
M 3 2
M 1 1
M 4 1
Q 4 4 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2
X 2
M 1 3
M 4 6
M 1 1
M 4 4
Q 12 2 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4
X 3
M 1 3
M 4 10
Q 12 4 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2
X 2
M 2 8
M 1 1
M 4 8
Q 2 4 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4
X 1
M 1 2
M 3 1
M 2 18
M 4 4
Q 11 3 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3
X 2
M 3 3
M 1 3
M 4 16
Q 3 4 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3
X 3
M 4 1
M 1 4
M 3 1
M 4 14
Q 5 2 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4
X 2
M 1 3
M 4 10
M 2 1
M 4 1
M 2 3
M 4 1
Q 6 2 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4
X 2
M 1 4
M 4 12
M 2 2
```

M 4 2
Q 3 1 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3
X 2
M 1 5
M 4 14
Q 10 3 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2
X 2
M 3 1
M 1 3
M 4 10
M 2 2
M 4 1
M 2 1
M 4 2
Q 9 2 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3
X 1
M 1 3
M 4 10
M 1 3
M 4 2
M 1 3
M 3 1
M 4 5
M 1 2
M 4 1
Q 11 4 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4
X 1
M 3 2
M 1 3
M 4 5
M 1 2
M 3 4
M 4 6
M 1 1
M 4 1
M 1 6
M 3 1
M 4 2
Q 5 2 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2
X 1
M 3 1
M 1 2
M 4 3
M 1 1
M 4 10
Q 4 4 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3
X 3
M 1 5
M 4 8
M 1 1
M 4 5
Q 9 3 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3
X 3
M 3 1
M 1 5
M 4 6
M 2 2
M 4 1
M 2 1

M 4 2
M 2 1
M 4 3
Q 11 3 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3
X 2
M 1 1
M 3 2
M 1 1
M 3 1
M 1 4
M 4 11
Q 4 4 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1
X 3
M 3 1
M 2 3
M 4 1
M 2 1
M 4 5
Q 10 4 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1
X 1
M 1 2
M 3 1
M 1 1
M 4 12
Q 5 3 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2
X 2
M 1 1
M 3 2
M 1 2
M 4 10
M 2 2
M 4 1
Q 1 2 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4
X 1
M 3 4
M 1 3
M 4 3
M 1 1
M 4 7
Q 5 3 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2
X 2
M 3 2
M 1 8
M 4 4
Q 3 4 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4
X 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 4
Q 4 2 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1
X 1
M 2 1
M 4 1
M 3 1
M 1 11
M 4 2
Q 6 3 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4

X 2
M 1 3
M 4 9
Q 7 3 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2
X 3
M 3 1
M 1 3
M 4 1
M 1 1
M 4 1
M 1 1
M 4 7
Q 4 3 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1
X 1
M 1 3
M 4 6
M 1 1
M 4 2
Q 7 1 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1
X 1
M 3 5
M 4 1
M 1 1
M 4 6
Q 1 1 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4
X 3
M 3 3
M 2 6
M 4 1
M 2 1
M 4 7
Q 10 2 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3
X 3
M 1 6
M 4 8
Q 8 4 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3
X 1
M 1 2
M 4 9
Q 10 2 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4
X 2
M 3 1
M 4 4
M 2 4
M 4 1
M 2 13
M 4 1
Q 5 4 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2
X 1
M 1 1
M 3 1
M 1 4
M 4 8
Q 2 1 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1
X 1
M 2 7
M 4 1
M 2 6
M 4 2

Q 10 4 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1
X 3
M 1 3
M 4 8
Q 7 2 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2
X 1
M 1 2
M 4 8
Q 5 1 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4
X 1
M 3 3
M 2 5
M 4 3
M 2 1
M 4 1
M 1 1
M 4 4
Q 10 1 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2
X 2
M 3 2
M 2 9
M 4 2
M 2 1
M 4 2
Q 1 4 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2
X 3
M 2 7
M 4 10
Q 10 3 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3
X 3
M 3 1
M 1 6
M 4 10
Q 7 3 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3
X 1
M 4 6
M 1 1
M 4 4
Q 6 4 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3
X 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 7
Q 2 2 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1
X 1
M 2 6
M 4 1
M 2 1
M 4 15
Q 11 1 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4
X 2
M 3 1
M 4 7
M 2 1
M 4 6
Q 8 1 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4

```
X 1
M 2 7
M 4 17
Q 1 2 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1
X 1
M 4 1
M 3 1
M 1 3
M 4 6
M 2 1
M 4 6
Q 2 4 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1
X 1
M 2 7
M 4 15
Q 3 2 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3
X 3
M 1 3
M 4 6
M 1 1
M 4 10
Q 12 2 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4
X 1
M 4 1
M 2 3
M 4 10
M 2 2
M 4 1
M 2 7
M 4 3
Q 1 3 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3
X 3
M 3 2
M 2 1
M 4 15
Q 2 3 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3
X 3
M 3 2
M 2 3
M 3 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 14
Q 2 3 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3
X 3
M 4 7
M 1 4
M 4 5
M 1 11
M 4 1
Q 5 1 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1
X 1
M 4 7
M 1 4
M 4 5
M 1 1
```

```
M 4 1
M 1 4
M 3 1
M 4 4
Q 6 4 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2
X 3
M 4 8
M 2 5
M 4 4
M 2 7
M 4 3
Q 3 4 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4
X 1
M 3 1
M 4 3
M 1 4
M 4 1
M 1 2
M 4 1
M 1 3
M 4 1
M 1 1
M 4 1
Q 3 1 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2
X 2
M 4 7
M 2 3
M 4 3
M 2 2
M 4 2
M 3 2
M 2 1
M 4 1
M 2 4
M 3 1
M 4 3
Q 12 1 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2
X 3
M 1 8
M 4 2
Q 5 3 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2
X 2
M 2 1
M 3 1
M 1 3
M 3 1
M 4 7
M 1 3
M 4 5
M 1 2
M 4 1
M 1 3
M 3 5
M 1 1
M 4 2
M 3 1
M 4 1
Q 12 4 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2
X 1
```

```
M 4 8
M 1 5
M 4 4
M 1 6
M 4 1
Q 9 3 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1
X 3
M 4 9
M 1 3
M 4 2
M 1 1
M 4 1
M 1 13
M 4 1
Q 7 3 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1
X 3
M 3 1
M 4 7
M 3 3
M 2 3
M 4 2
M 2 1
M 4 1
M 2 1
M 4 1
M 1 1
M 4 6
Q 5 3 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3
X 2
M 2 2
M 3 1
M 4 7
M 3 1
M 4 2
M 2 2
M 4 1
M 2 1
M 4 1
M 1 5
M 4 6
Q 11 1 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2
X 3
M 3 1
M 1 2
M 4 6
M 1 2
M 4 6
M 1 7
M 4 2
Q 5 2 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2
X 2
M 3 3
M 2 1
M 4 11
M 2 1
M 4 1
M 2 1
M 4 2
M 1 1
```

```
M 4 1
M 2 1
M 3 1
M 2 5
M 3 3
M 2 1
M 4 2
Q 3 4 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3
X 3
M 3 1
M 4 9
M 1 2
M 4 7
Q 5 2 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2
X 1
M 3 3
M 1 5
M 4 17
Q 12 2 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4
X 1
M 4 4
M 1 6
M 4 1
M 1 2
M 4 1
Q 4 2 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1
X 3
M 2 2
M 4 5
M 2 3
M 3 1
M 4 6
M 2 1
M 4 2
M 2 6
M 3 1
M 2 5
M 4 1
Q 9 2 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4
X 3
M 1 3
M 4 11
M 1 3
M 4 1
M 1 3
M 4 2
Q 9 1 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3
X 1
M 4 7
M 1 3
M 4 2
M 1 1
M 4 1
M 1 1
M 4 2
M 1 10
M 4 1
Q 4 1 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2
X 3
```

M 2 1
M 4 8
M 1 4
M 4 1
M 1 1
M 4 7
Q 4 3 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1
X 3
M 2 2
M 4 8
M 2 1
M 4 1
M 1 1
M 4 7
M 1 1
M 4 1
Q 11 2 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4
X 3
M 2 2
M 4 7
M 2 1
M 4 4
M 2 1
M 4 1
M 2 13
M 4 1
Q 11 2 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3
X 2
M 3 3
M 4 11
M 2 5
M 4 2
M 2 8
M 4 1
M 2 1
M 4 1
Q 10 3 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4
X 1
M 4 5
M 2 3
M 4 7
M 2 7
M 4 1
M 3 1
M 4 1
Q 4 2 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3
X 2
M 2 3
M 3 1
M 4 17
Q 12 4 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3
X 2
M 2 3
M 4 8
M 2 1
M 4 4
Q 6 1 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4
X 2
M 2 6

M 4 6
M 2 3
M 4 4
M 2 1
M 4 3
M 2 1
M 4 3
Q 1 4 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4
X 3
M 4 3
M 3 1
M 4 5
M 1 1
M 4 1
M 1 1
M 4 7
Q 1 3 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3
X 3
M 2 3
M 4 7
M 2 1
M 4 1
M 2 1
M 4 7
Q 4 2 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3
X 2
M 3 1
M 4 11
M 2 1
M 4 1
M 1 1
M 4 3
Q 11 1 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2
X 1
M 2 1
M 4 9
M 2 1
M 3 2
M 4 5
Q 12 4 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2
X 1
M 4 7
M 2 3
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 3
M 4 1
Q 5 3 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1
X 2
M 3 3
M 2 3
M 4 14

Q 2 4 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4
X 1
M 1 6
M 4 15
Q 5 3 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1
X 1
M 3 2
M 1 2
M 4 11
M 1 1
M 4 5
Q 2 3 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4
X 1
M 3 1
M 2 6
M 4 7
M 2 1
M 4 8
Q 11 4 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2
X 2
M 3 3
M 1 2
M 3 3
M 1 2
M 4 11
M 1 1
M 4 5
Q 2 4 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2
X 1
M 4 6
M 1 1
M 3 1
M 1 4
M 4 10
Q 8 3 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1
X 2
M 4 8
M 2 6
M 4 4
M 2 5
M 4 4
Q 8 3 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4
X 1
M 4 6
M 1 6
M 2 9
M 4 3
M 2 1
M 4 1
M 2 1
M 4 1
M 2 1
M 4 1
M 2 5
M 4 3
Q 12 2 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2
X 2
M 1 5
M 4 5

M 1 4
M 4 1
Q 5 2 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3
X 3
M 2 1
M 3 1
M 4 7
M 2 3
M 4 4
M 2 1
M 4 1
M 2 8
M 4 1
Q 4 1 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4
X 2
M 3 1
M 4 7
M 1 2
M 4 2
M 3 1
M 2 6
M 4 1
M 2 1
M 4 7
Q 1 4 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2
X 2
M 3 3
M 1 1
M 4 16
Q 5 1 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1
X 1
M 3 2
M 2 5
M 4 11
M 2 1
M 4 5
Q 6 4 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4
X 3
M 4 11
M 2 1
M 4 5
Q 6 2 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4
X 2
M 4 8
M 2 2
M 4 1
M 2 1
M 4 7
Q 11 2 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1
X 3
M 3 3
M 1 5
M 4 5
M 1 2
M 4 1
M 1 3
M 4 5
Q 12 1 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2
X 2

```
M 4 8
M 2 6
M 4 4
M 2 1
M 4 1
M 2 3
M 4 1
Q 10 4 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3
X 3
M 4 6
M 3 1
M 4 3
M 2 6
M 4 3
M 2 1
M 4 1
M 2 3
M 3 1
M 4 1
M 2 1
M 4 3
Q 7 2 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1
X 2
M 3 1
M 4 7
M 1 5
M 4 5
M 1 7
M 4 1
Q 11 3 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1
X 2
M 3 1
M 4 17
Q 3 4 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1
X 2
M 3 1
M 1 2
M 4 16
Q 7 2 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3
X 3
M 4 1
M 3 3
M 1 4
M 4 12
M 1 1
M 4 4
Q 11 1 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4
X 2
M 4 7
M 1 3
M 4 1
M 1 1
M 4 3
M 1 1
M 4 6
Q 1 4 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4
X 2
M 3 3
M 4 8
```

M 1 2
M 4 8
Q 8 4 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4
X 1
M 2 6
M 4 5
M 2 8
M 4 2
Q 7 1 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3
X 2
M 4 7
M 2 6
M 4 10
Q 10 2 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4
X 2
M 3 1
M 4 3
M 2 5
M 4 1
M 2 1
M 4 1
M 2 6
M 4 1
Q 10 3 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2
X 3
M 1 5
M 4 5
M 1 3
M 4 1
M 3 1
M 1 1
M 4 3
Q 7 1 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2
X 3
M 3 2
M 4 12
M 1 1
M 4 3
Q 2 1 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2
X 1
M 4 4
M 1 2
M 4 1
M 1 14
M 4 1
Q 4 1 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4
X 1
M 4 4
M 1 5
M 4 1
M 1 7
M 4 1
Q 9 3 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4 4 1
X 1
M 3 1
M 4 8
M 2 8
M 4 4
M 2 2

```
M 4 1
M 2 1
M 4 1
0
Q 10 4 11 4 4 4 10 3 7 4 3 2 9 2 8 2 2 4 4 1 6 1
X 1
```


1112


1112